Física Experimental VI – 4300314

2º Semestre de 2023

Instituto de Física
Universidade de São Paulo

Professor: Antonio Domingues dos Santos

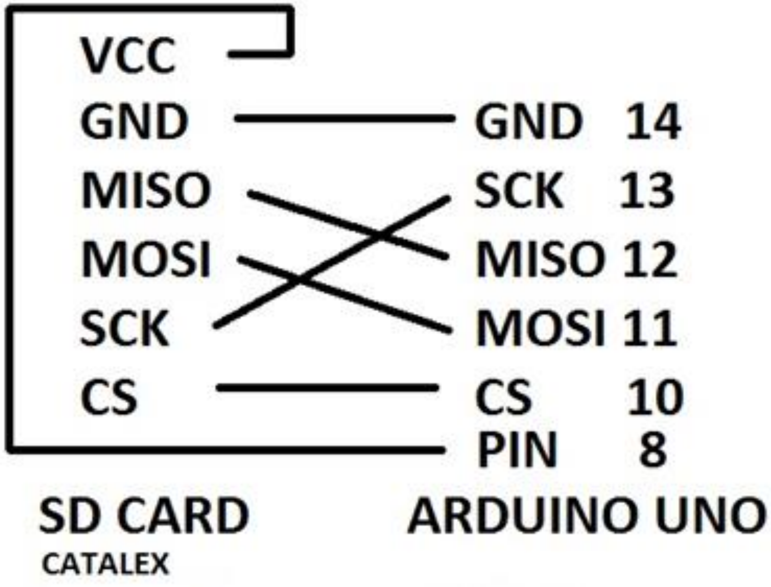
E-mail: adsantos@if.usp.br
Fone: 3091.6886

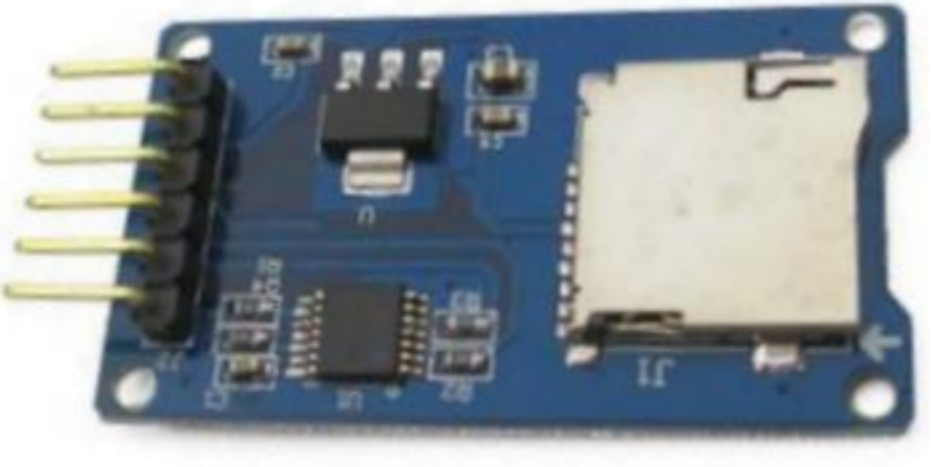
Atenção

Os dados neste arquivo, sobre sensores, transdutores e outros dispositivos para Arduíno, foram obtidos ao longo dos últimos anos e podem não estar atualizados.

Arduino - cartão microSD

Protocolo de comunicação SPI



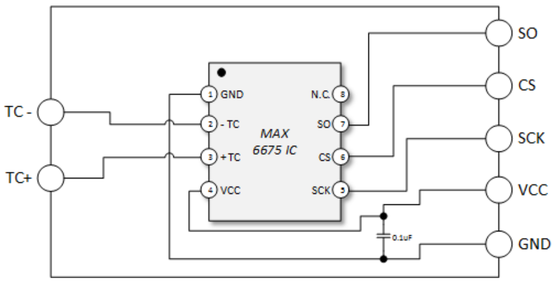


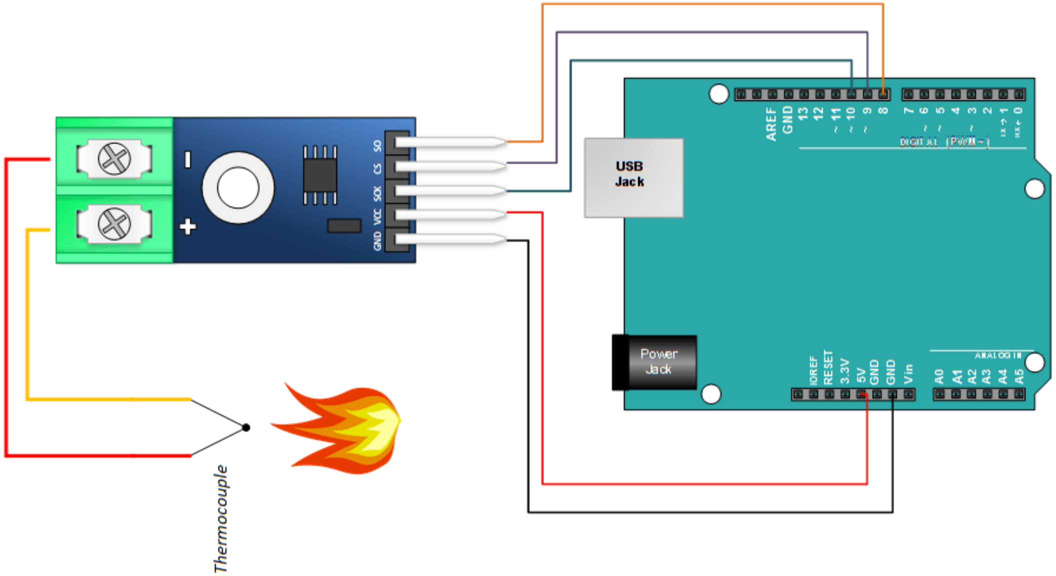
Cartões microSD podem ser obtidos em celulares antigos.

https://www.arduino.cc/en/reference/SD

Arduino – Termopar do tipo K e módulo Max6675

Protocolo de comunicação SPI





Supply Voltage	3.3. to 5 VDC
Operating Current	about 50mA
Measurement Range	0 to 1024 deg C (32 deg F to 1875 F)
Measurement Resolution	+/- 0.25 Deg C (+/- 0.45 Deg F)
Output	Uses a SPI Interface
Required SENSOR	K Thermocouple

SO: The module's serial output.   Your Arduino will read this output.

CS: Chip Select.   Setting low, selects the Module and tells it to supply an output that is synchronize with a clock.

SCK: The Serial Clock… an input from your Arduino.

VCC: 5V supply.

GND: Ground.

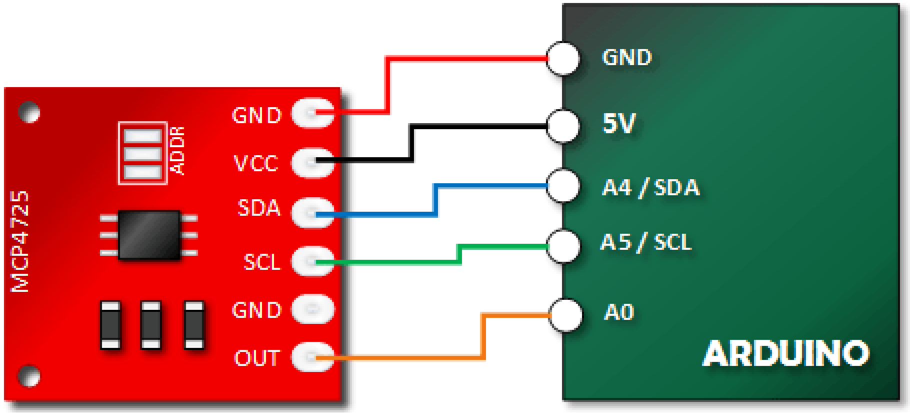
– :   The K thermocouple minus input (Az).

+ : The K Thermocouple plus input (Vm).

Um pino para cada unidade.

http://henrysbench.capnfatz.com/henrys-bench/arduino-temperature-measurements/max6675-temp-module-arduino-manual-and-tutorial/

Arduino – Conversor Digital-Analógico
DAC 12 bits MCP4725
Protocolo de comunicação I^2C

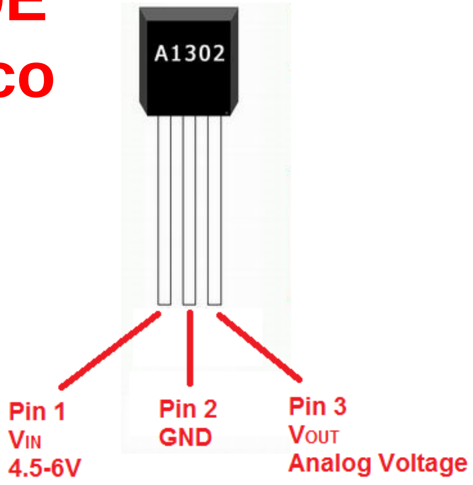


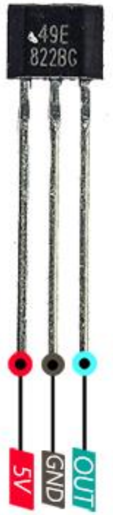
- CI: MCP4725;
- Tensão de operação: 2.7-5.5V;
- Resolução: 12-bit;
- Interface: I2C;
- Endereço I2C: 0x62 (pino A0 - LOW)
ou 0x63 (pino A0 - HIGH).

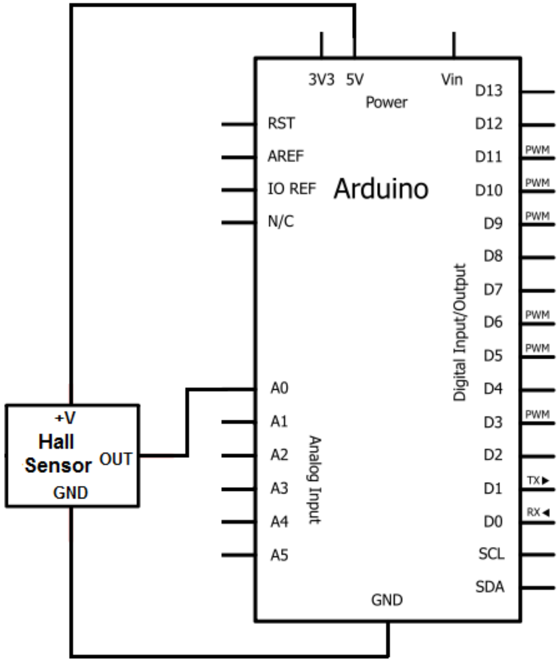
http://henrysbench.capnfatz.com/henrys-bench/arduino-output-devices/arduino-mcp4725-digital-to-analog-converter-tutorial/

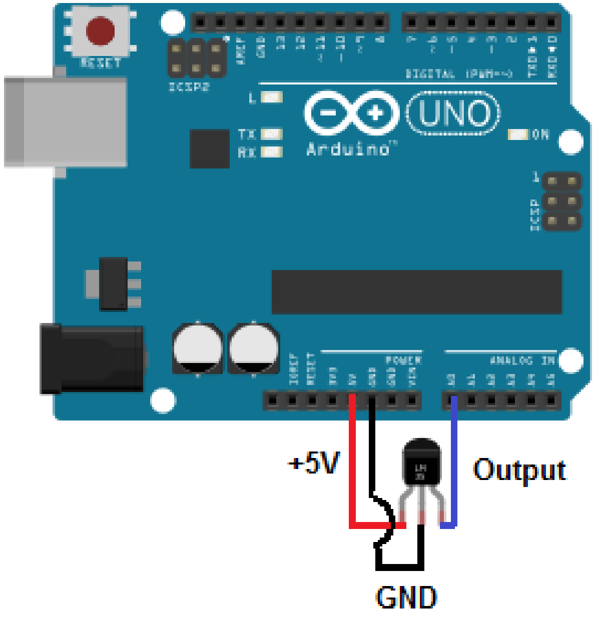
Arduino – Sensor Hall A1302 ou SS49E
Medida de campo magnético

Usar entrada analógica







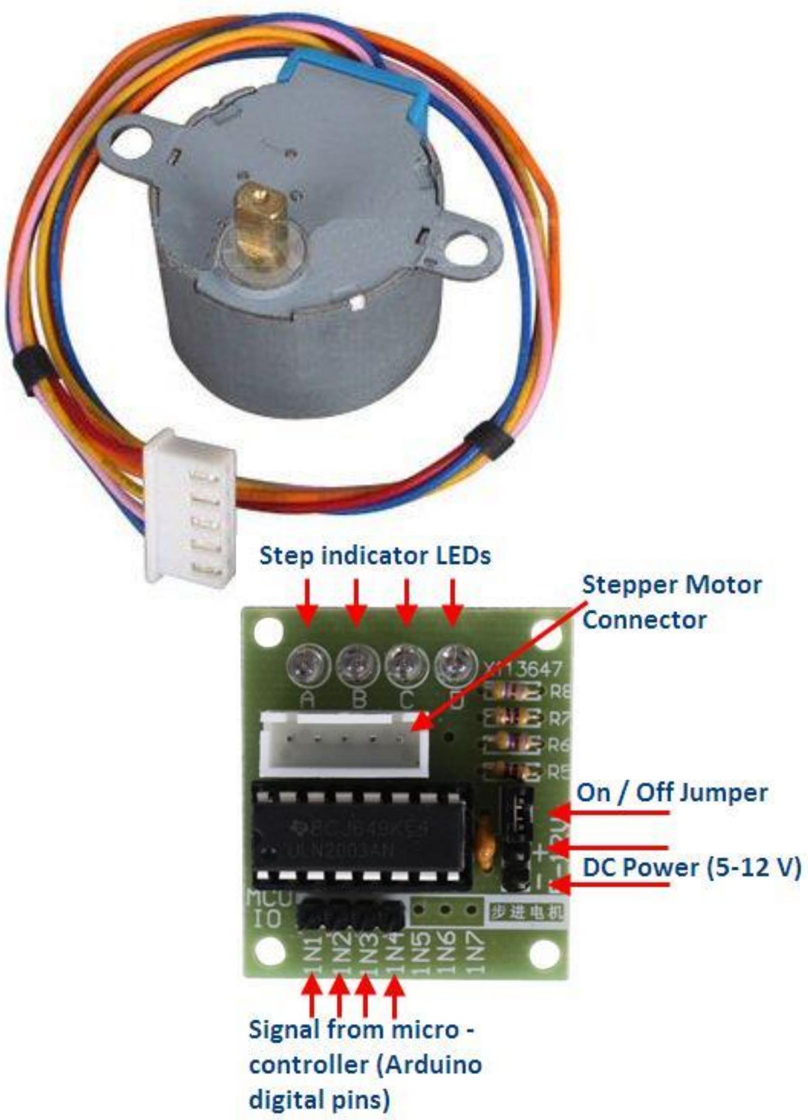


A sensibilidade do A1302 é 1,3 mV/G e do SS49E é 1,8 (ou 1,4 !) mV/G.
Atenção, 0 G corresponde a $V_{cc}/2$.

http://www.learningaboutelectronics.com/Articles/Hall-effect-sensor-circuit.php /

Arduino – Motor de passo com driver ULN2003

Saídas digitais

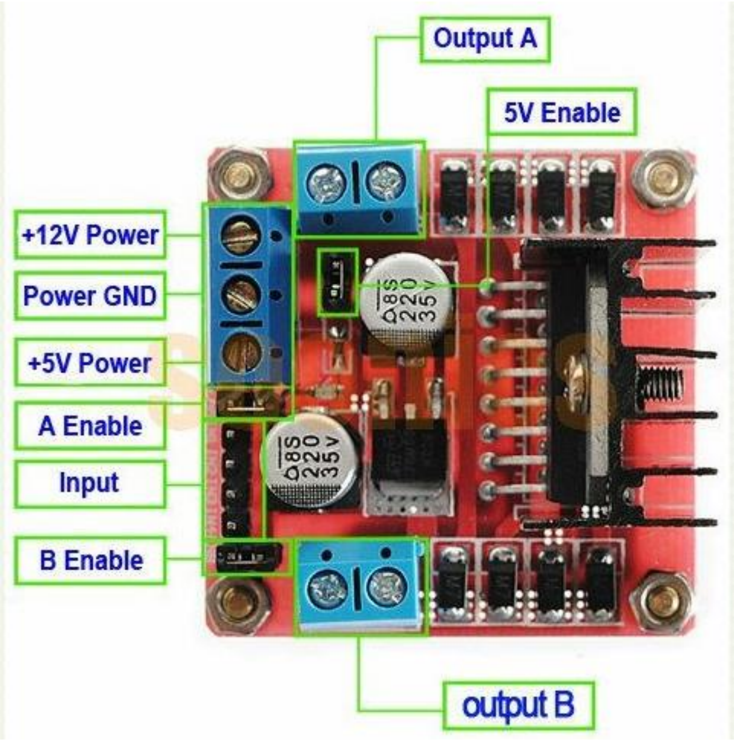


Motor Type Unipolar stepper motor
Connection Type5 Wire Connection (to the motor controller)
Voltage 5-12 Volts DC
Frequency 100 Hz
Step mode Half-step mode recommended (8 step control signal sequence)
Step angle **Half-step mode: 8 step control signal sequence (recommended)** 5.625 degrees per step / 64 steps per one revolution of the internal motor shaft**Full Step mode: 4 step control signal sequence** 11.25 degrees per step / 32 steps per one revolution of the internal motor shaft. Gear ratio is 63.684:1

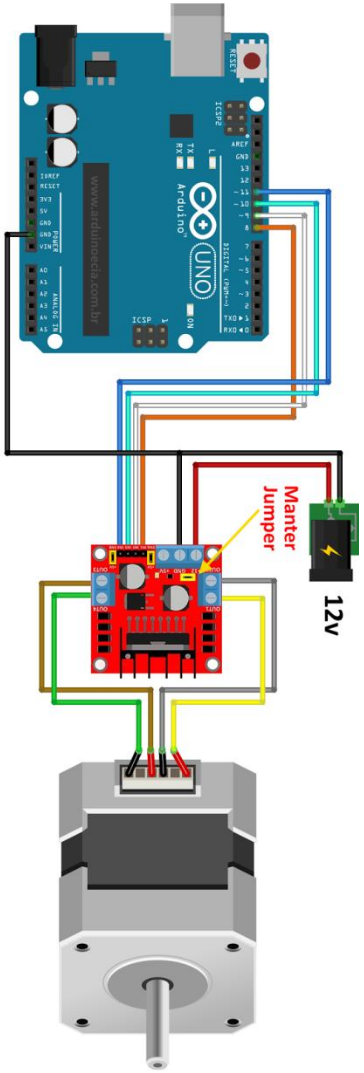
http://42bots.com/tutorials/28byj-48-stepper-motor-with-uln2003-driver-and-arduino-uno/

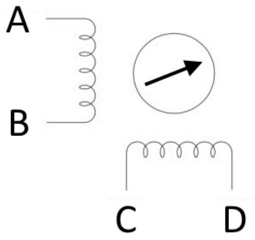
Arduino – Motor dc com driver L298n

Saídas digitais



The L298N H-bridge module can be used with motors that have a voltage of between 5 and 35V DC. With the module used in this tutorial, there is also an onboard 5V regulator, so if your supply voltage is up to 12V you can also source 5V from the board.





		Bobinas			
		A	B	C	D
Passos	1	1	0	0	1
	2	0	1	0	1
	3	0	1	1	0
	4	1	0	1	0

Control also a Stepper Motor with Arduino and L298N

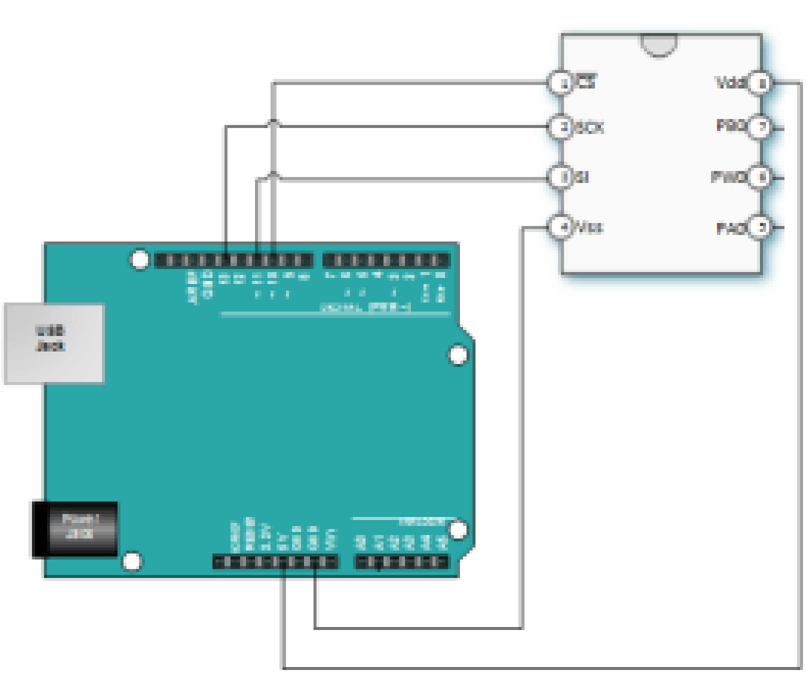
http://www.arduinoecia.com.br/2014/08/ponte-h-l298n-motor-de-passo.html

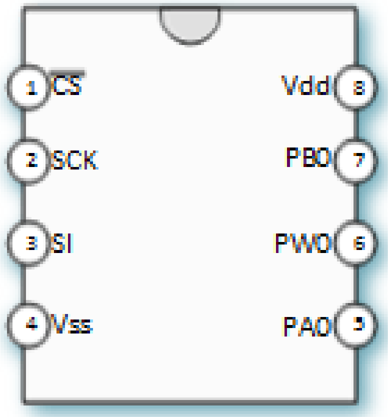
http://www.instructables.com/id/Control-DC-and-stepper-motors-with-L298N-Dual-Moto/

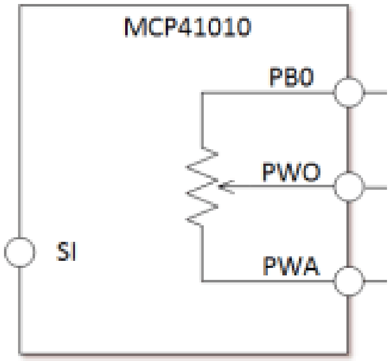
Arduino – Potenciômetro Digital – MCP41010

(256 níveis em um total de 10 kΩ)

Protocolo de comunicação SPI







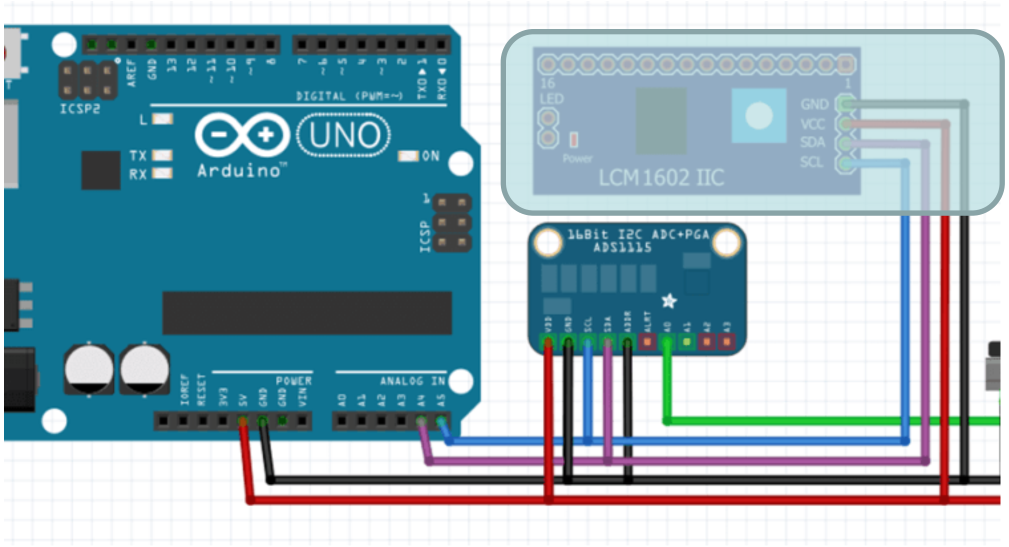
Pin 1 – CS – Chip Select. When low, the chip will receive commands from serial Input at pin 3.

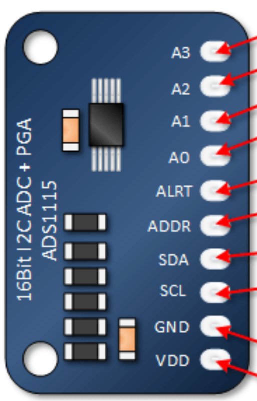
Pin 2 – SCK – Serial Clock. Clock input from the micro-controller that synchronizes serial communications. (porta 13 do Arduino Uno)

Pin 3 – SI – Serial Input. Receives commands from the micro-controller when CS at pin 1 is low. (porta 11 do Arduino Uno)

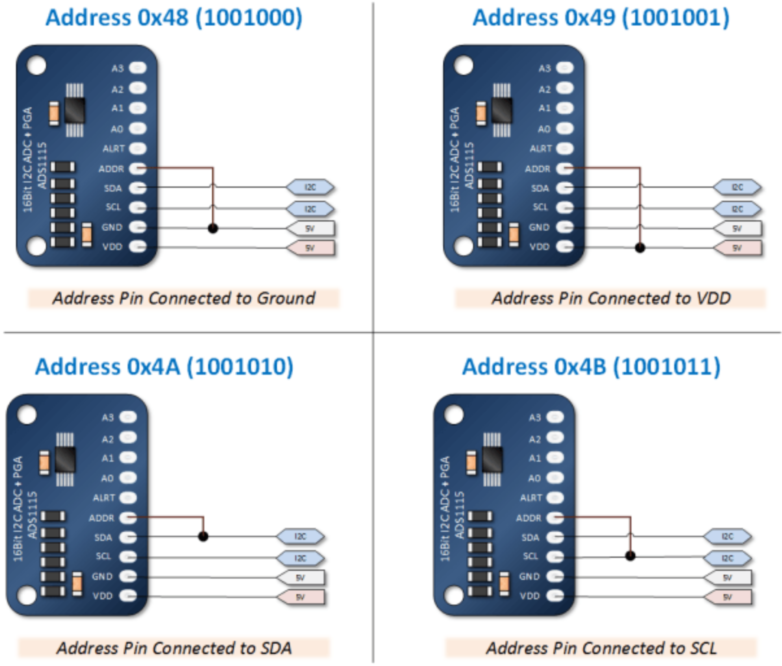
http://henrysbench.capnfatz.com/henrys-bench/arduino-output-devices/mcp41010-digital-potentiometer-arduino-user-manual/

Arduino – ADC + PGA (12 bits/4canais) ADS1115 (Adafruit)
Protocolo de comunicação I²C
Conversor Analógico-Digital





A3 – Analog Input 3
A2 – Analog Input 2
A1 – Analog Input 1
A0 – Analog Input 0
ALRT – Alert/Ready
ADDR – Address Pin
SDA – I2C SDA
SCL – I2C SCL
GND – Ground In
VDD – 2.0 to 5.5V Supply



Address 0x48 (1001000)

Address Pin Connected to Ground

Address 0x49 (1001001)

Address Pin Connected to VDD

Address 0x4A (1001010)

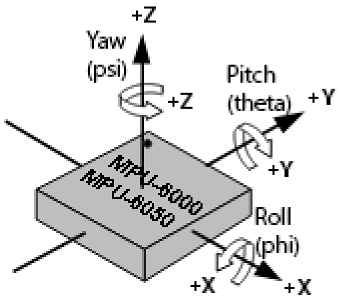
Address Pin Connected to SDA

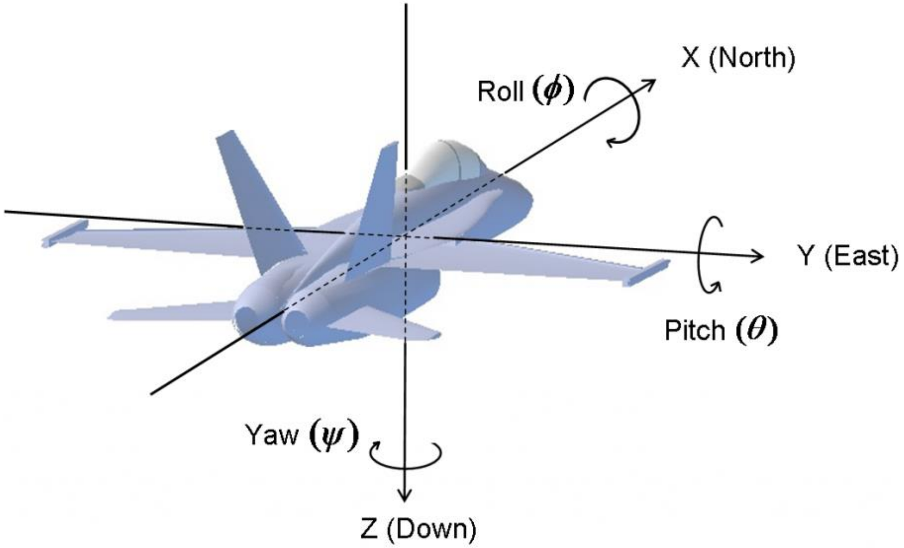
Address 0x4B (1001011)

Address Pin Connected to SCL

http://henrysbench.capnfatz.com/henrys-bench/arduino-voltage-measurements/arduino-ads1115-module-getting-started-tutorial/

Arduino – MPU6050 (3+3 axis) Acelerômetro e Giroscópio
Protocolo de comunicação I^2C



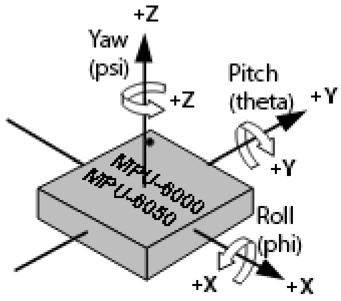


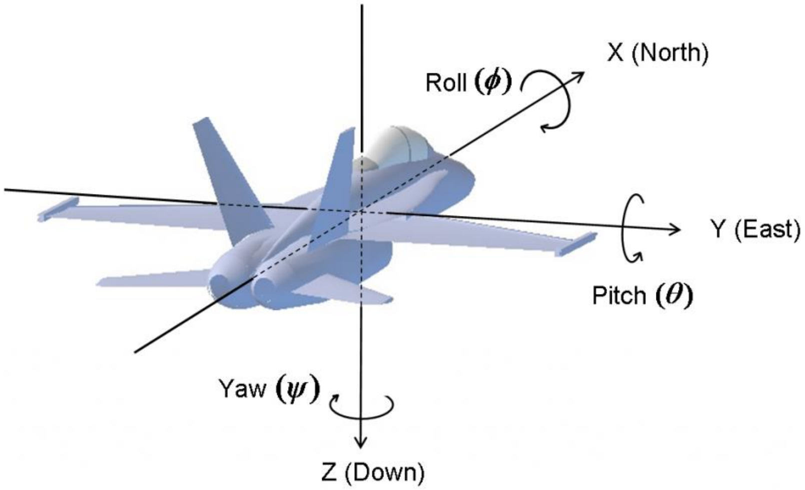
MPU6050 → Pin ID
VDD → 5V
GND → GND
SCL → SCL (A5)
SDA → SDA (A4)
XDA
XCL
ADO → GND
INT

Ver MPU9250= MPU6050 + Campo magnético

Jeff Rowberg jeff@rowberg.net https://github.com/jrowberg/i2cdevlib/arduino/mpu6050

Arduino – MPU6050 (3+3 axis) Acelerômetro e Giroscópio Protocolo de comunicação I²C

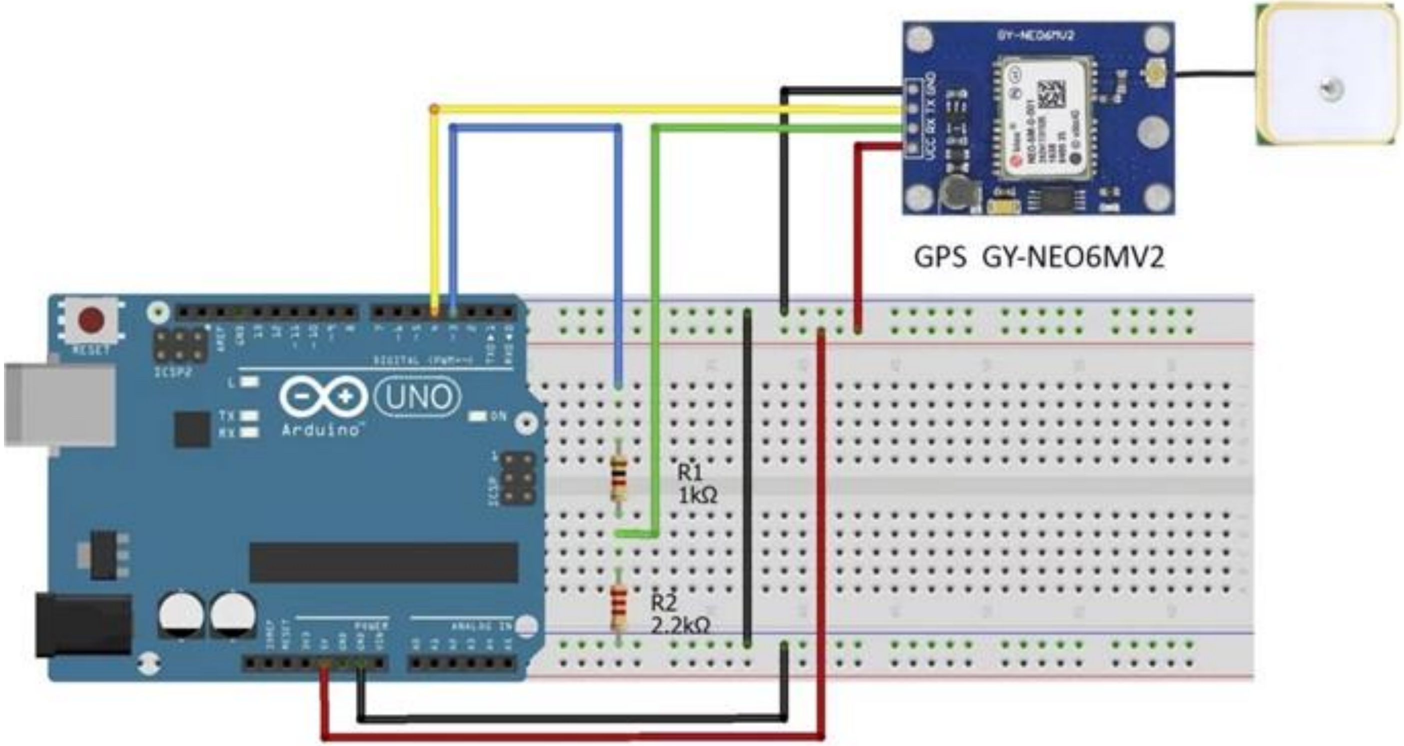




```
// Librerias I2C para controlar el mpu6050
// la libreria MPU6050.h necesita I2Cdev.h, I2Cdev.h necesita Wire.h
#include "I2Cdev.h“
#include "MPU6050.h“
#include "Wire.h“
// La dirección del MPU6050 puede ser 0x68 o 0x69, dependiendo
// del estado de AD0. Si no se especifica, 0x68 estará implicito
MPU6050 sensor;
// Valores RAW (sin procesar) del acelerometro y giroscopio en los
ejes x,y,z
int ax, ay, az;int gx, gy, gz;

void setup() {
Serial.begin(57600);    //Iniciando puerto serial
Wire.begin();           //Iniciando I2C
sensor.initialize();    //Iniciando el sensor
if (sensor.testConnection()) Serial.println("Sensor iniciado
correctamente");
  else Serial.println("Error al iniciar el sensor");
}

void loop() {      // Leer las aceleraciones y velocidades angulares
sensor.getAcceleration(&ax, &ay, &az);
sensor.getRotation(&gx, &gy, &gz);
//Mostrar las lecturas separadas por un [tab]
Serial.print("a[x y z] g[x y z]:\t");  Serial.print(ax); Serial.print("\t");
Serial.print(ay); Serial.print("\t");  Serial.print(az); Serial.print("\t");
Serial.print(gx); Serial.print("\t");  Serial.print(gy); Serial.print("\t");
Serial.println(gz);
delay(100);
}
```

Arduino – GPS Neo-6M (GPS6MV2)
Comunicação pela porta serial, em 3,3V.

```
#include <SoftwareSerial.h>
SoftwareSerial gps(4,3);
char dados= ' ‘;

void setup() {
 Serial.begin(115200);
 gps.begin(9600);
 }

void loop() {
  if(gps.available())   {
    dados=gps.read();
    Serial.print(dados);
    delay (50);
    }
 }
```



Arduino – GPS Neo-6M (GPS6MV2), com library TinyGPS

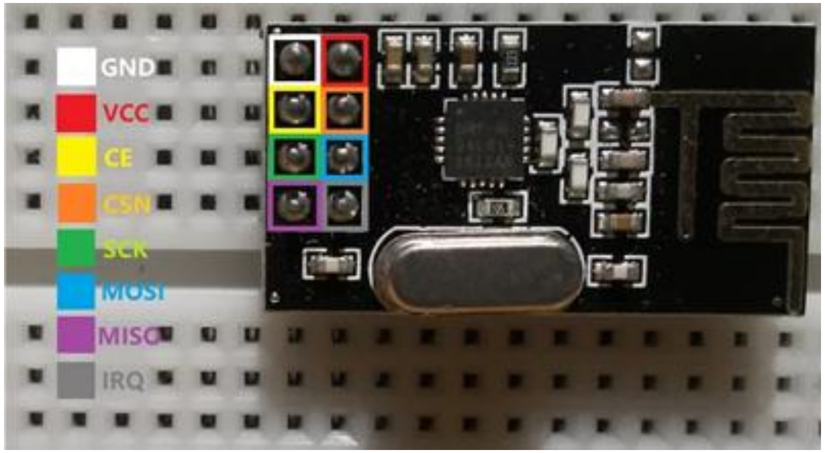
https://blog.eletrogate.com/gps-neo-6m-com-arduino-aprenda-usar/

```
#include <SoftwareSerial.h>//incluimos SoftwareSerial
#include <TinyGPS.h>//incluimos TinyGPS

TinyGPS gps;//Declaramos el objeto gps
SoftwareSerial serialgps(4,3);//Declaramos el pin 4 Tx y 3 Rx
//Declaramos la variables para la obtención de datos
int year;
byte month, day, hour, minute, second, hundredths;
unsigned long chars;
unsigned short sentences, failed_checksum;

void setup()
{

Serial.begin(115200);//Iniciamos el puerto serie
serialgps.begin(9600);//Iniciamos el puerto serie del gps
//Imprimimos:
Serial.println("");
Serial.println("GPS GY-GPS6MV2 Leantec");
Serial.println(" ---Buscando senal--- ");
Serial.println("");
}
```

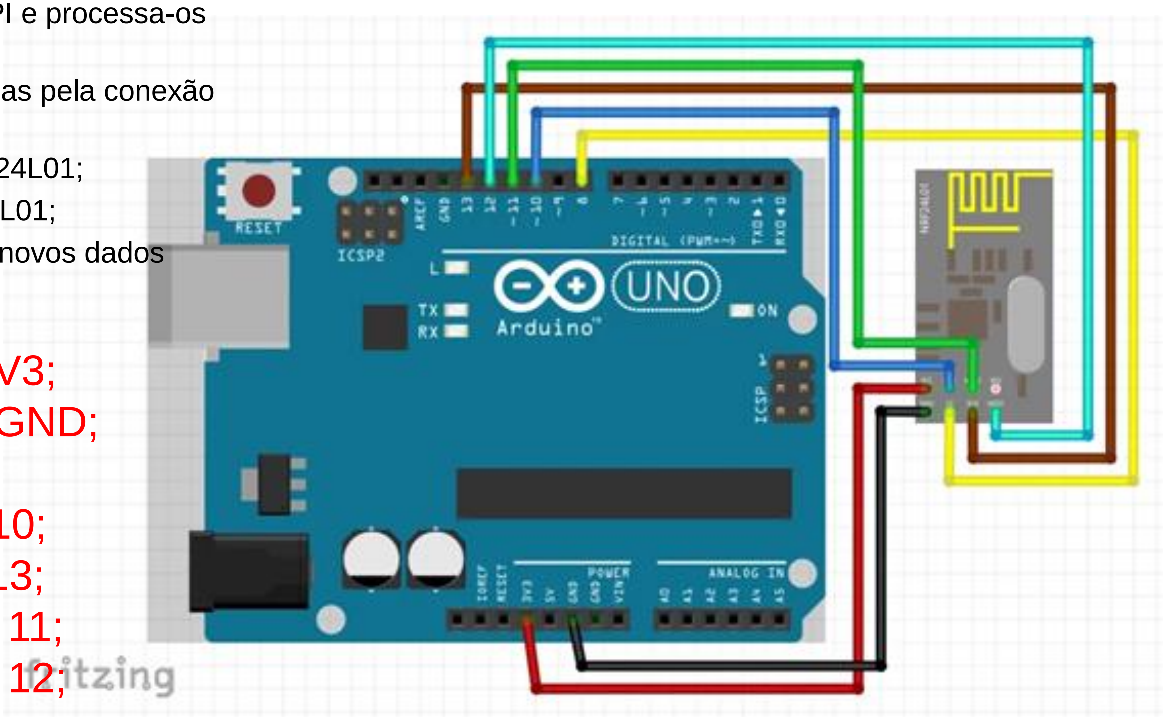
```
void loop()
{
while(serialgps.available())
{
int c = serialgps.read();
if(gps.encode(c))
{
float latitude, longitude;
gps.f_get_position(&latitude, &longitude);
Serial.print("Latitud/Longitud: ");
Serial.print(latitude,5);
Serial.print(", ");
Serial.println(longitude,5);
gps.crack_datetime(&year,&month,&day,&hour,&minute,&second,&hundredths);
Serial.print("Fecha: "); Serial.print(day, DEC); Serial.print("/");
Serial.print(month, DEC); Serial.print("/"); Serial.print(year);
Serial.print(" Hora: "); Serial.print(hour, DEC); Serial.print(":");
Serial.print(minute, DEC); Serial.print(":"); Serial.print(second, DEC);
Serial.print("."); Serial.println(hundredths, DEC);
Serial.print("Altitud (metros): "); Serial.println(gps.f_altitude());
Serial.print("Rumbo (grados): "); Serial.println(gps.f_course());
Serial.print("Velocidad(kmph): ");
Serial.println(gps.f_speed_kmph());
Serial.print("Satelites: "); Serial.println(gps.satellites());
Serial.println();
gps.stats(&chars, &sentences, &failed_checksum);
}
}
}
```

Arduino – Módulo de comunicação wireless NRF24L01

- **GND**: é o pino Ground. Geralmente, é marcado por um quadrado branco ao seu redor, assim podendo ser utilizado como referência para os outros pinos;
- **VCC**: fornece energia para o módulo. Ele pode ser de 1,9 V até 3,9 V. Pode ser conectado a saída de 3,3 V do Arduino. ATENÇÃO: Conectar o módulo NRF24L01+ a saída de 5 V do Arduino pode queimá-lo!
- **CE (Chip Enable)**: é um pino ativamente alto (active-HIGH). Quando selecionado, o módulo irá transmitir ou receber, dependendo do modo em que estiver.
- **CSN (Chip Select Not)**: é um pino ativamente baixo (active-LOW) porém é geralmente mantido em HIGH. Quando este pino estiver em LOW, o NRF24L01 começa a captar dados da porta SPI e processa-os de acordo com suas especificações.
- **SCK (Serial Clock)**: entrada de pulsos de clock providas pela conexão Mestre do SPI;
- **MOSI (Master Out Slave In)**: entrada SPI para o NRF24L01;
- **MISO (Master In Slave Out)**: saída SPI para o NRF24L01;
- **IRQ**: é um pino interruptor que avisa o Mestre quando novos dados estão disponíveis para serem processados.



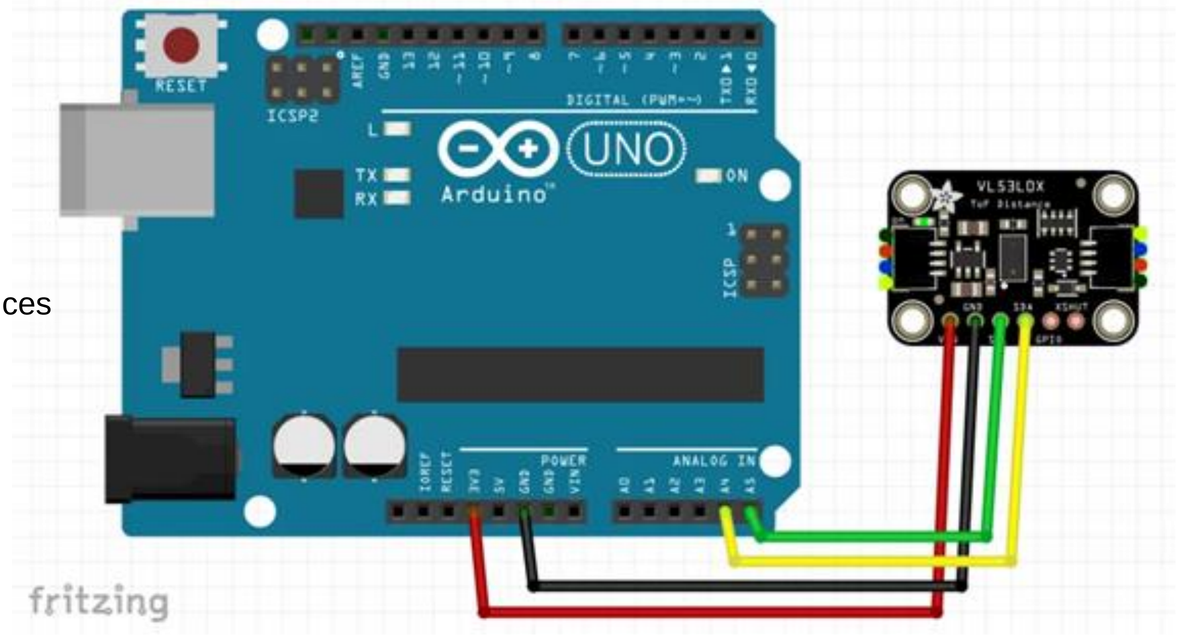
Vcc -> 3V3;
GND -> GND;
CE -> 8;
CSN -> 10;
SCK -> 13;
MOSI -> 11;
MISO -> 12;



https://blog.eletrogate.com/guia-definitivo-do-modulo-wireless-nrf24l01-2/

Arduino – Módulo Lidar VL53L0X

Library   Adafruit_VL53L0X



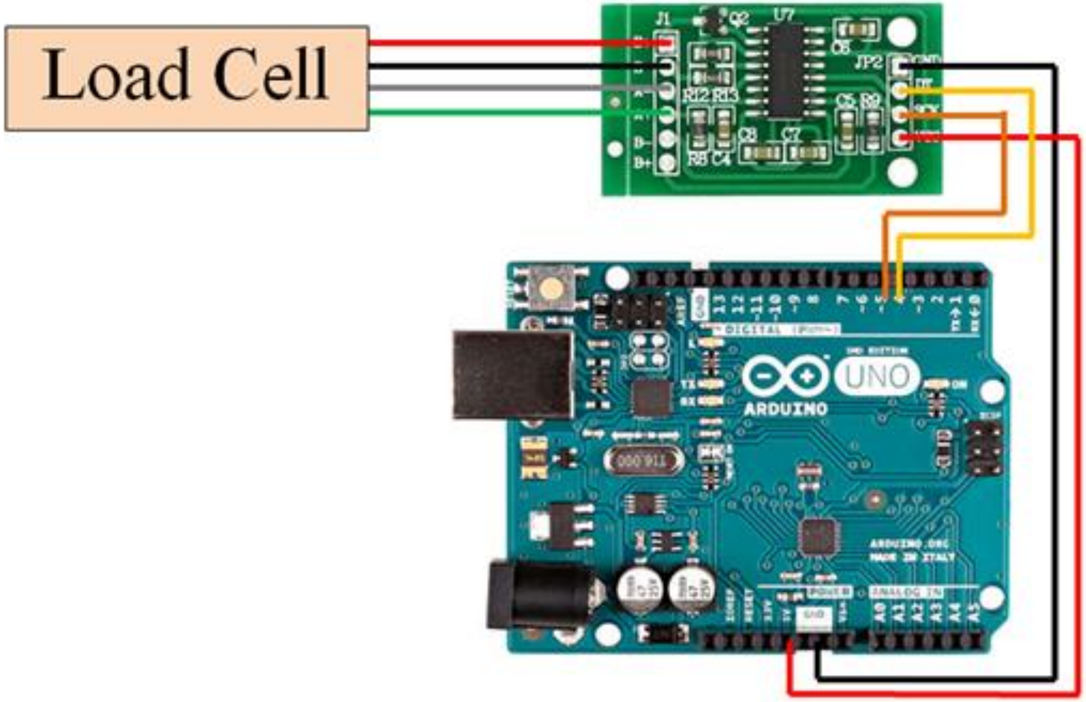
```
#include "Adafruit_VL53L0X.h"
Adafruit_VL53L0X lox = Adafruit_VL53L0X();

void setup() {
  Serial.begin(115200);
// wait until serial port opens for native USB devices
  while (! Serial) {
    delay(1);
  }
  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
// power
  Serial.println(F("VL53L0X API Simple Ranging example\n\n"));
}

void loop() {
  VL53L0X_RangingMeasurementData_t measure;
  Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!
  if (measure.RangeStatus != 4) {    // phase failures have incorrect data
    Serial.print("Distance (mm): "); Serial.println(measure.RangeMilliMeter);
  } else {
    Serial.println(" out of range ");
  }
  delay(100);
}
```

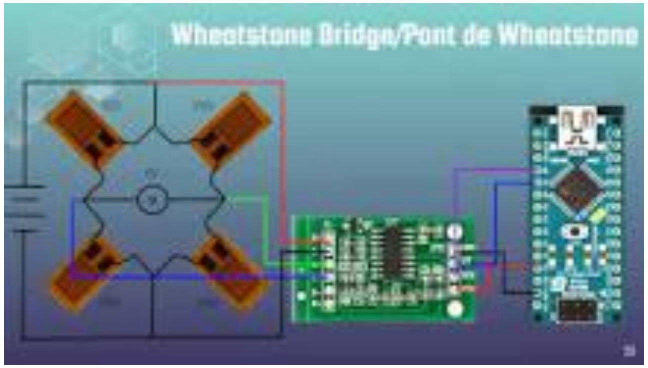
I2C
Vcc -> 3V3;
GND -> GND;
SCL -> A5;
SDA -> A4;
GPI01 ->
XSHUT ->

https://blog.eletrogate.com/sensor-de-distancia-laser-gy-vl53l0xv2/

Arduino – Módulo HX711 e StrainGauge

Library HX711
https://github.com/bogde/HX711





```
#include "HX711.h"
HX711 loadcell;

// 1. HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 2;
const int LOADCELL_SCK_PIN = 3;

// 2. Adjustment settings
const long LOADCELL_OFFSET = 50682624;
const long LOADCELL_DIVIDER = 5895655;

// 3. Initialize library
loadcell.begin(LOADCELL_DOUT_PIN,
LOADCELL_SCK_PIN);
loadcell.set_scale(LOADCELL_DIVIDER);
loadcell.set_offset(LOADCELL_OFFSET);

// 4. Acquire reading
Serial.print("Weight: ");
Serial.println(loadcell.get_units(10), 2);
```

SPI
Vcc -> 5V
GND -> GND
SCL -> 3
SDA -> 2

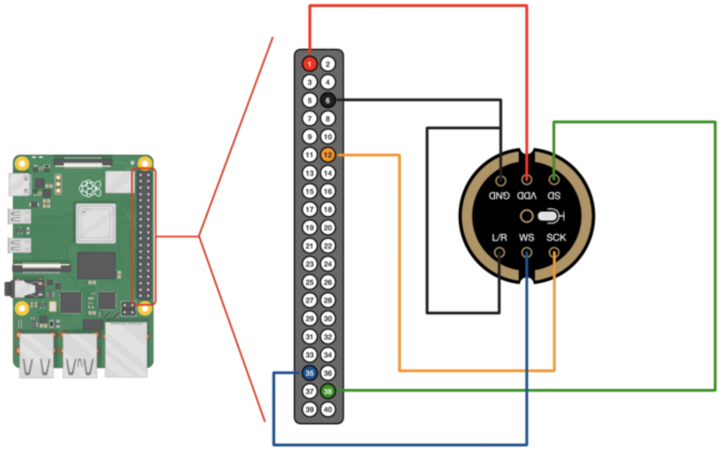
https://pt.slideshare.net/KarineSilva26/colagem-de-strain-gages-em-clula-de-carga

Arduino – Microfone INMP441

Library I2S

https://github.com/makerportal/rpi_i2s

https://forum.arduino.cc/t/decaler-un-signal/994290/19



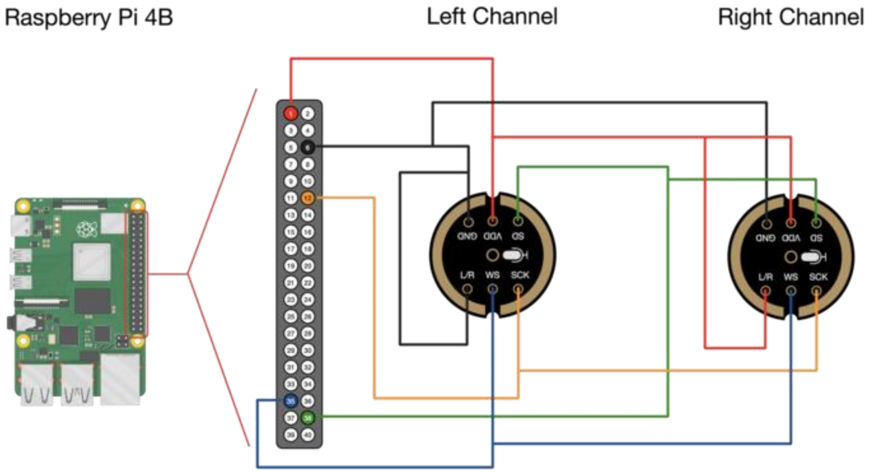
ArduinoSound Library

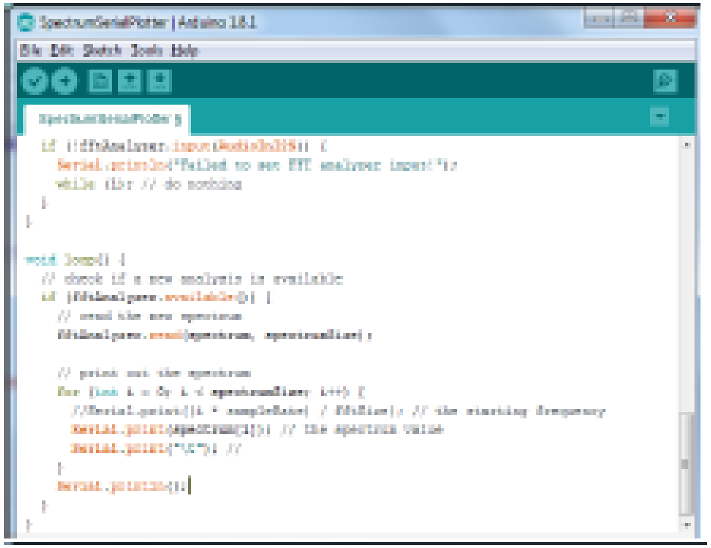
For most uses, its better to have a higher-level library for managing sound. The `ArduinoSound` library works with I2S mics and can do filtering, amplitude detection, etc!

Install it using the Arduino library manager

Various examples come with the library, check them out in the File->Examples->ArduinoSound sub menu

You can also do FFT spectral diagramming using SpectrumSerialPlotter.




SpectrumSerialPlotter | Arduino 1.8.1
File Edit Sketch Tools Help
SpectrumSerialPlotter §

```
  if (!fftAnalyzer.input(AudioInI2S)) {
    Serial.println("Failed to set FFT analyzer input!");
    while (1); // do nothing
  }
}

void loop() {
  // check if a new analysis is available
  if (fftAnalyzer.available()) {
    // read the new spectrum
    fftAnalyzer.read(spectrum, spectrumSize);

    // print out the spectrum
    for (int i = 0; i < spectrumSize; i++) {
      //Serial.print((i * sampleRate) / fftSize); // the starting frequency
      Serial.print(spectrum[i]); // the spectrum value
      Serial.print("\t"); //
    }
    Serial.println();
  }
}
```

https://makersportal.com/blog/recording-stereo-audio-on-a-raspberry-pi

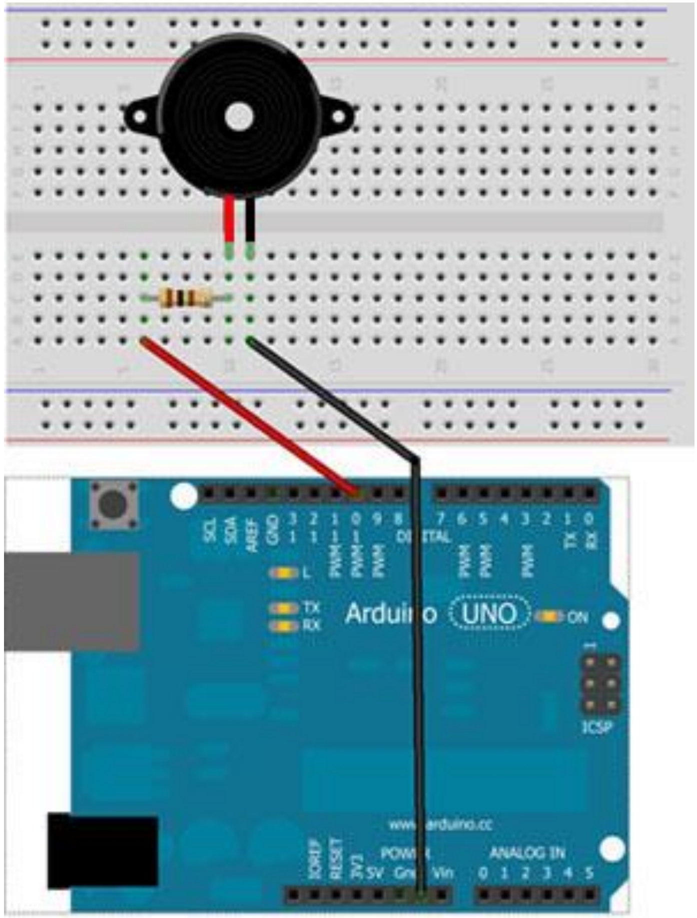
Arduino – Buzzer

Active Buzzer

```
int buzzerPin = 9
int buttonPin = 7;

void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  }

void loop() {
  int buttonState = digitalRead(buttonPin);
  If (buttonState == LOW) {
    digitalWrite(buzzerPin, HIGH);
    }
  If (buttonState == HIGH) {
    digitalWrite(buzzerPin, LOW);
    }
  }
```

Passive Buzzer

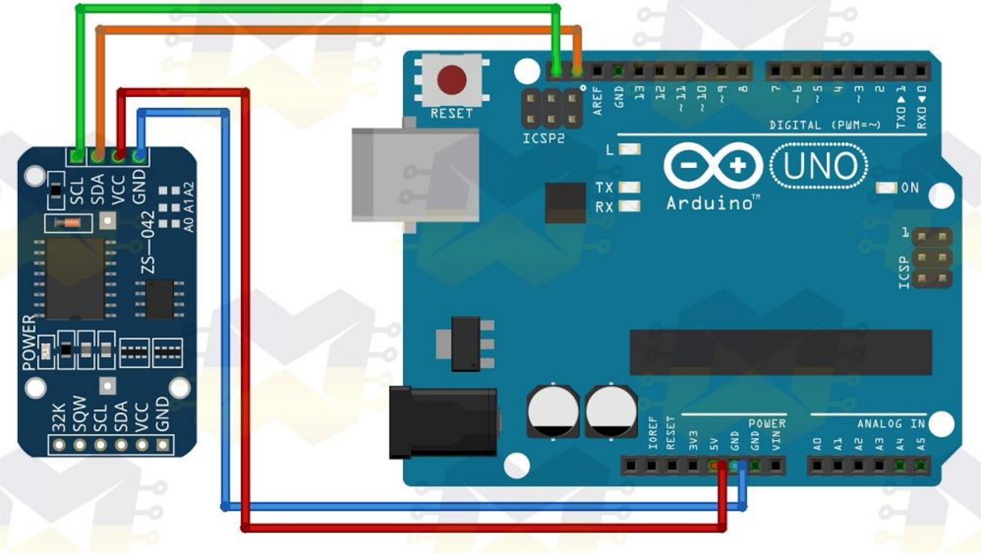
```
int buzzerPin = 8;

void setup() {
  pinMode(buzzerPin, OUTPUT);
  tone(buzzerPin, 1000, 2000);
  }

void loop() {
  tone(buzzerPin, 440); // A4
  delay(1000);
  tone(buzzerPin, 494); // B4
  delay(1000);
  tone(buzzerPin, 523); // C4
  delay(1000);
  tone(buzzerPin, 587); // D4
  delay(1000);
  tone(buzzerPin, 659); // E4
  delay(1000);
  tone(buzzerPin, 698); // F4
  delay(1000);
  tone(buzzerPin, 784); // G4
  delay(1000);
  noTone(buzzerPin);
  delay(1000);
  }
```



Made with Fritzing.org

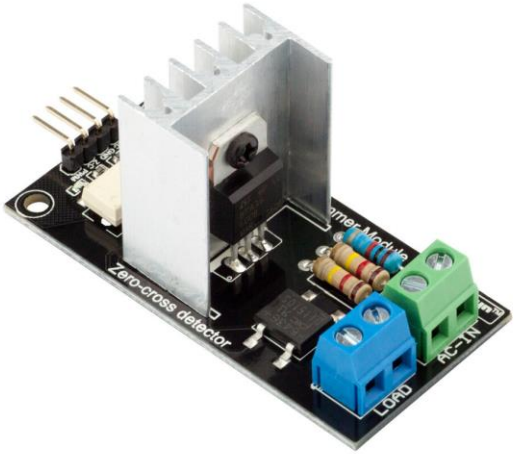
Arduino – Módulo de Relógio RTC DS1307

```
•#include <Wire.h>      //INCLUSÃO DA BIBLIOTECA
•#include "RTClib.h“    //INCLUSÃO DA BIBLIOTECA
• RTC_DS3231 rtc;      //OBJETO DO TIPO RTC_DS3231
• //DECLARAÇÃO DOS DIAS DA SEMANA
•char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"};
•
•void setup(){
•  Serial.begin(9600);
•  if(! rtc.begin()) {         // SE O RTC NÃO FOR INICIALIZADO, FAZ
•    Serial.println("DS3231 não encontrado");
•    while(1);                 //SEMPRE ENTRE NO LOOP
•  }
•  if(rtc.lostPower()){        //SE RTC FOI LIGADO PELA PRIMEIRA VEZ / FICOU SEM ENERGIA / ESGOTOU A BATERIA, FAZ
•    Serial.println("DS3231 OK!");
•    //REMOVA O COMENTÁRIO DE UMA DAS LINHAS ABAIXO PARA INSERIR AS INFORMAÇÕES ATUALIZADAS EM SEU RTC
•    //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));       //CAPTURA A DATA E HORA EM QUE O SKETCH É COMPILADO
•    //rtc.adjust(DateTime(2018, 9, 29, 15, 00, 45));         //(ANO), (MÊS), (DIA), (HORA), (MINUTOS), (SEGUNDOS)
•  }
•  delay(100);
•}
•
•void loop () {
•    DateTime now = rtc.now();       //CHAMADA DE FUNÇÃO
•    Serial.print("Data: ");
•    Serial.print(now.day(), DEC);    //IMPRIME O DIA
•    Serial.print('/');
•    Serial.print(now.month(), DEC);     //IMPRIME O MÊS
•    Serial.print('/');
•    Serial.print(now.year(), DEC);       //IMPRIME O ANO
•    Serial.print(" / Dia: ");
•    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);     //IMPRIME O DIA
•    Serial.print(" / Horas: ");
•    Serial.print(now.hour(), DEC);       //IMPRIME A HORA
•    Serial.print(':');
•    Serial.print(now.minute(), DEC);       //IMPRIME OS MINUTOS
•    Serial.print(':');
•    Serial.print(now.second(), DEC);       //IMPRIME OS SEGUNDOS
•    Serial.println();
•    delay(1000);
•}
```



https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-real-time-clock-rtc-ds3231

Arduino – Dimmer Module (5A)



```
/**************
 *  RobotDyn
 *  Dimmer Library
 *  **************
 *
 *  The following sketch is meant to define dimming value through potentiometer,
* /

#include <RBDdimmer.h>//

//#define USE_SERIAL  SerialUSB //Serial for boards whith USB serial port
#define USE_SERIAL  Serial
#define outputPin  12
#define zerocross  2

//dimmerLamp dimmer(outputPin, zerocross); //initialase port for dimmer for ESP8266, ESP32, Arduino due boards
dimmerLamp dimmer(outputPin); //initialase port for dimmer for MEGA, Leonardo, UNO, Arduino M0, Arduino Zero

int outVal = 0;

void setup() {
  USE_SERIAL.begin(9600);
  dimmer.begin(NORMAL_MODE, ON); //dimmer initialisation: name.begin(MODE, STATE)
}

void loop()
{
  outVal = map(analogRead(0), 1, 1024, 100, 0); // analogRead(analog_pin), min_analog, max_analog, 100%, 0%);
  USE_SERIAL.println(outVal);
  dimmer.setPower(outVal); // name.setPower(0%-100%)
}
```

https://github.com/RobotDynOfficial/RBDDimmer

Arduino – Plotter

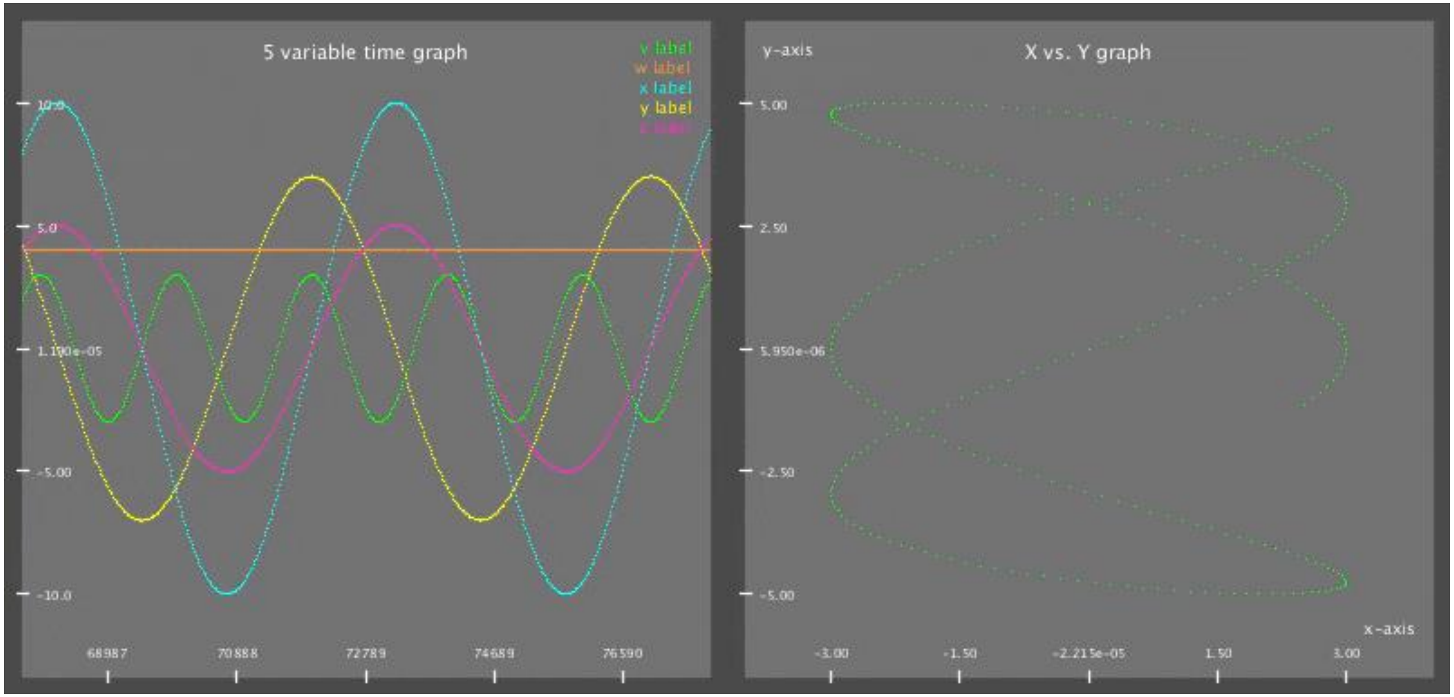
An Arduino library for easy plotting on host computer via serial communication
by: Devin Conley_
COPY OF REPOSITORY FOR ARDUINO LIBRARY MANAGER
For more information, quick-start guide, documentation and listeners, please go to:
https://github.com/devinaconley/arduino-plotter

1) Add a multi-variable graph vs. time

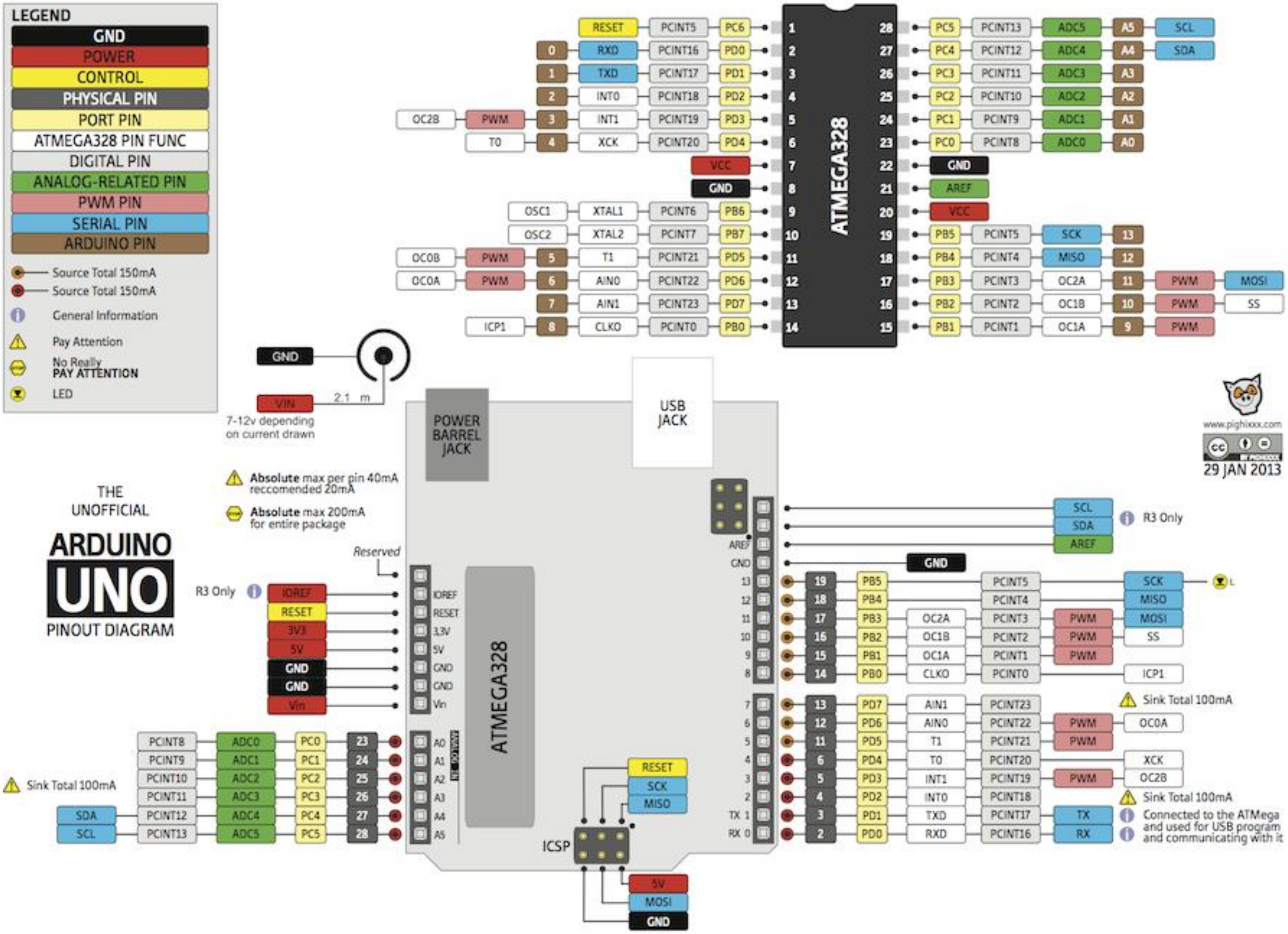
void AddTimeGraph(String title, int pointsDisplayed, String label1, Variable1Type variable1, String label2, Variable2Type variable2, ...)

2) Add an X vs Y graph

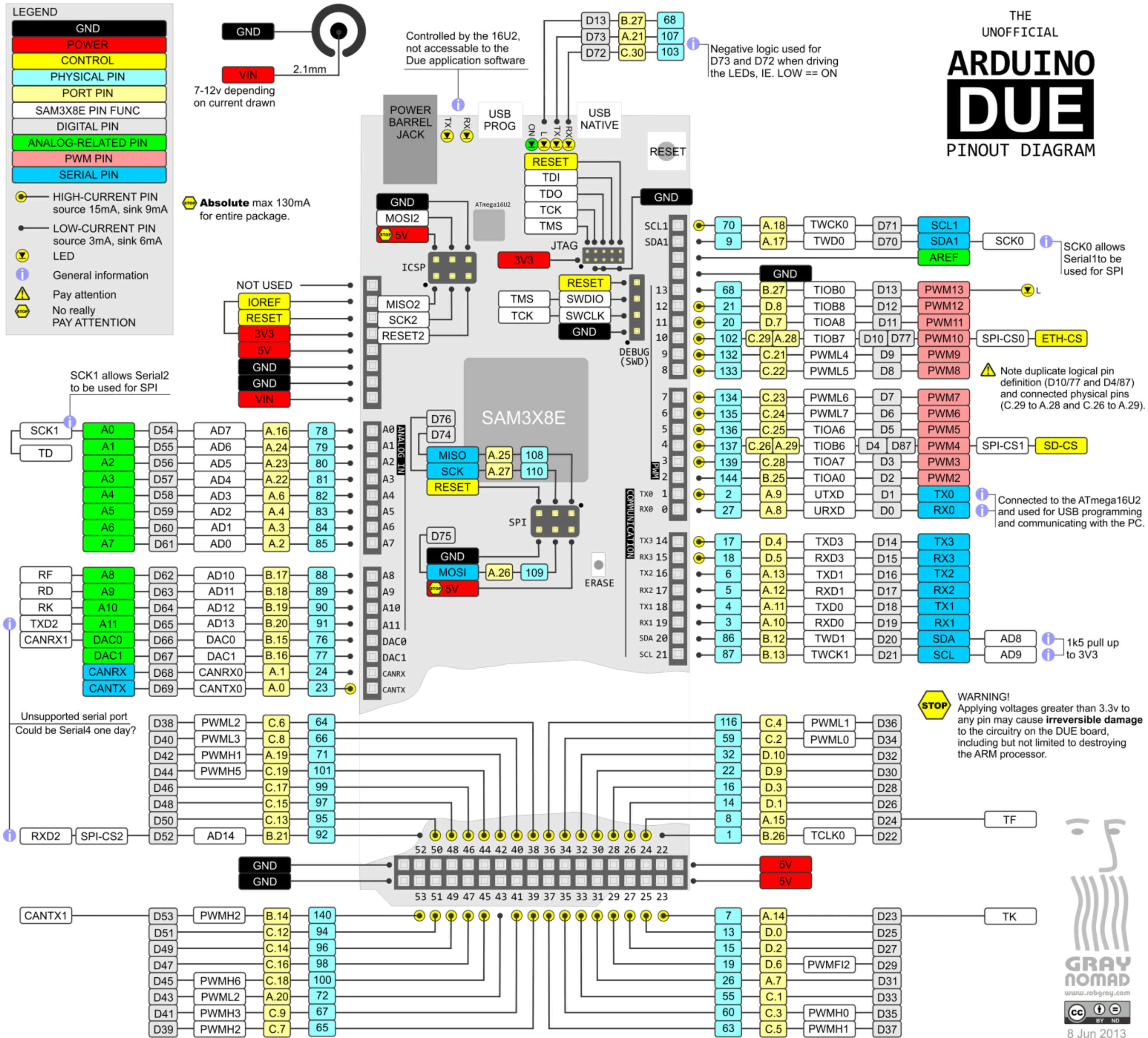
void AddXYGraph(String title, int pointsDisplayed, String labelX, VariableTypeX variableX, String labelY, VariableTypeY variableY)



Arduino – UNO


LEGEND
GND
POWER
CONTROL
PHYSICAL PIN
PORT PIN
ATMEGA328 PIN FUNC
DIGITAL PIN
ANALOG-RELATED PIN
PWM PIN
SERIAL PIN
ARDUINO PIN
Source Total 150mA
Source Total 150mA
General Information
Pay Attention
No Really PAY ATTENTION
LED
RESET PCINT5 PC6 1
0 RXD PCINT16 PD0 2
1 TXD PCINT17 PD1 3
2 INT0 PCINT18 PD2 4
OC2B PWM 3 INT1 PCINT19 PD3 5
T0 4 XCK PCINT20 PD4 6
VCC 7
GND 8
OSC1 XTAL1 PCINT6 PB6 9
OSC2 XTAL2 PCINT7 PB7 10
OC0B PWM 5 T1 PCINT21 PD5 11
OC0A PWM 6 AIN0 PCINT22 PD6 12
7 AIN1 PCINT23 PD7 13
ICP1 8 CLKO PCINT0 PB0 14
ATMEGA328
28 PC5 PCINT13 ADC5 A5 SCL
27 PC4 PCINT12 ADC4 A4 SDA
26 PC3 PCINT11 ADC3 A3
25 PC2 PCINT10 ADC2 A2
24 PC1 PCINT9 ADC1 A1
23 PC0 PCINT8 ADC0 A0
22 GND
21 AREF
20 VCC
19 PB5 PCINT5 SCK 13
18 PB4 PCINT4 MISO 12
17 PB3 PCINT3 OC2A 11 PWM MOSI
16 PB2 PCINT2 OC1B 10 PWM SS
15 PB1 PCINT1 OC1A 9 PWM
GND
VIN
2.1 m
7-12v depending on current drawn
POWER BARREL JACK
USB JACK
www.pighixxx.com
29 JAN 2013
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
THE UNOFFICIAL
ARDUINO
UNO
PINOUT DIAGRAM
Reserved
R3 Only
IOREF
RESET
3V3
5V
GND
GND
Vin
IOREF
RESET
3.3V
5V
GND
GND
Vin
ATMEGA328
ANALOG IN
PCINT8 ADC0 PC0 23 A0
PCINT9 ADC1 PC1 24 A1
PCINT10 ADC2 PC2 25 A2
PCINT11 ADC3 PC3 26 A3
SDA PCINT12 ADC4 PC4 27 A4
SCL PCINT13 ADC5 PC5 28 A5
Sink Total 100mA
SCL
SDA
AREF
R3 Only
AREF
GND
GND
13 19 PB5 PCINT5 SCK L
12 18 PB4 PCINT4 MISO
11 17 PB3 OC2A PCINT3 PWM MOSI
10 16 PB2 OC1B PCINT2 PWM SS
9 15 PB1 OC1A PCINT1 PWM
8 14 PB0 CLKO PCINT0 ICP1
Sink Total 100mA
7 13 PD7 AIN1 PCINT23
6 12 PD6 AIN0 PCINT22 PWM OC0A
5 11 PD5 T1 PCINT21 PWM
4 6 PD4 T0 PCINT20 XCK
3 5 PD3 INT1 PCINT19 PWM OC2B
2 4 PD2 INT0 PCINT18
TX 1 3 PD1 TXD PCINT17 TX
RX 0 2 PD0 RXD PCINT16 RX
Sink Total 100mA
Connected to the ATMega and used for USB program and communicating with it
ICSP
RESET
SCK
MISO
5V
MOSI
GND

Arduino – Due



THE UNOFFICIAL
ARDUINO DUE
PINOUT DIAGRAM

LEGEND
- GND
- POWER
- CONTROL
- PHYSICAL PIN
- PORT PIN
- SAM3X8E PIN FUNC
- DIGITAL PIN
- ANALOG-RELATED PIN
- PWM PIN
- SERIAL PIN

HIGH-CURRENT PIN source 15mA, sink 9mA
LOW-CURRENT PIN source 3mA, sink 6mA
LED
General information
Pay attention
No really PAY ATTENTION

GND
VIN
2.1mm
7-12v depending on current drawn

Absolute max 130mA for entire package.

Controlled by the 16U2, not accessable to the Due application software

POWER BARREL JACK
TX RX
USB PROG
ON L TX RX
USB NATIVE
RESET

D13	B.27	68
D73	A.21	107
D72	C.30	103

Negative logic used for D73 and D72 when driving the LEDs, IE. LOW == ON

GND
MOSI2
5V
ICSP
ATmega16U2

RESET
TDI
TDO
TCK
TMS
JTAG
3V3
GND

SCL1	70	A.18	TWCK0	D71	SCL1	
SDA1	9	A.17	TWD0	D70	SDA1	SCK0
					AREF	
				GND		
13	68	B.27	TIOB0	D13	PWM13	L
12	21	D.8	TIOB8	D12	PWM12	
11	20	D.7	TIOA8	D11	PWM11	
10	102	C.29 A.28	TIOB7	D10 D77	PWM10	SPI-CS0 ETH-CS
9	132	C.21	PWML4	D9	PWM9	
8	133	C.22	PWML5	D8	PWM8	
7	134	C.23	PWML6	D7	PWM7	
6	135	C.24	PWML7	D6	PWM6	
5	136	C.25	TIOA6	D5	PWM5	
4	137	C.26 A.29	TIOB6	D4 D87	PWM4	SPI-CS1 SD-CS
3	139	C.28	TIOA7	D3	PWM3	
2	144	B.25	TIOA0	D2	PWM2	
TX0 1	2	A.9	UTXD	D1	TX0	
RX0 0	27	A.8	URXD	D0	RX0	
TX3 14	17	D.4	TXD3	D14	TX3	
RX3 15	18	D.5	RXD3	D15	RX3	
TX2 16	6	A.13	TXD1	D16	TX2	
RX2 17	5	A.12	RXD1	D17	RX2	
TX1 18	4	A.11	TXD0	D18	TX1	
RX1 19	3	A.10	RXD0	D19	RX1	
SDA 20	86	B.12	TWD1	D20	SDA	AD8
SCL 21	87	B.13	TWCK1	D21	SCL	AD9

SCK0 allows Serial1to be used for SPI

Note duplicate logical pin definition (D10/77 and D4/87) and connected physical pins (C.29 to A.28 and C.26 to A.29).

Connected to the ATmega16U2 and used for USB programming and communicating with the PC.

1k5 pull up to 3V3

PWM
COMMUNICATION

RESET
SWDIO
SWCLK
GND
TMS
TCK
DEBUG (SWD)

NOT USED
IOREF
RESET
3V3
5V
GND
GND
VIN

MISO2
SCK2
RESET2

SAM3X8E

D76
D74

MISO	A.25	108
SCK	A.27	110
RESET		

SPI

D75

GND		
MOSI	A.26	109
5V		

ERASE

SCK1 allows Serial2 to be used for SPI

ANALOG IN

SCK1	A0	D54	AD7	A.16	78	A0
TD	A1	D55	AD6	A.24	79	A1
	A2	D56	AD5	A.23	80	A2
	A3	D57	AD4	A.22	81	A3
	A4	D58	AD3	A.6	82	A4
	A5	D59	AD2	A.4	83	A5
	A6	D60	AD1	A.3	84	A6
	A7	D61	AD0	A.2	85	A7
RF	A8	D62	AD10	B.17	88	A8
RD	A9	D63	AD11	B.18	89	A9
RK	A10	D64	AD12	B.19	90	A10
TXD2	A11	D65	AD13	B.20	91	A11
CANRX1	DAC0	D66	DAC0	B.15	76	DAC0
	DAC1	D67	DAC1	B.16	77	DAC1
	CANRX	D68	CANRX0	A.1	24	CANRX
	CANTX	D69	CANTX0	A.0	23	CANTX

Unsupported serial port
Could be Serial4 one day?

		D38	PWML2	C.6	64
		D40	PWML3	C.8	66
		D42	PWMH1	A.19	71
		D44	PWMH5	C.19	101
		D46		C.17	99
		D48		C.15	97
		D50		C.13	95
RXD2	SPI-CS2	D52	AD14	B.21	92

52 50 48 46 44 42 40 38 36 34 32 30 28 26 24 22

GND GND

5V 5V

53 51 49 47 45 43 41 39 37 35 33 31 29 27 25 23

CANTX1	D53	PWMH2	B.14	140
	D51		C.12	94
	D49		C.14	96
	D47		C.16	98
	D45	PWMH6	C.18	100
	D43	PWML2	C.20	72
	D41	PWMH3	C.9	67
	D39	PWMH2	C.7	65

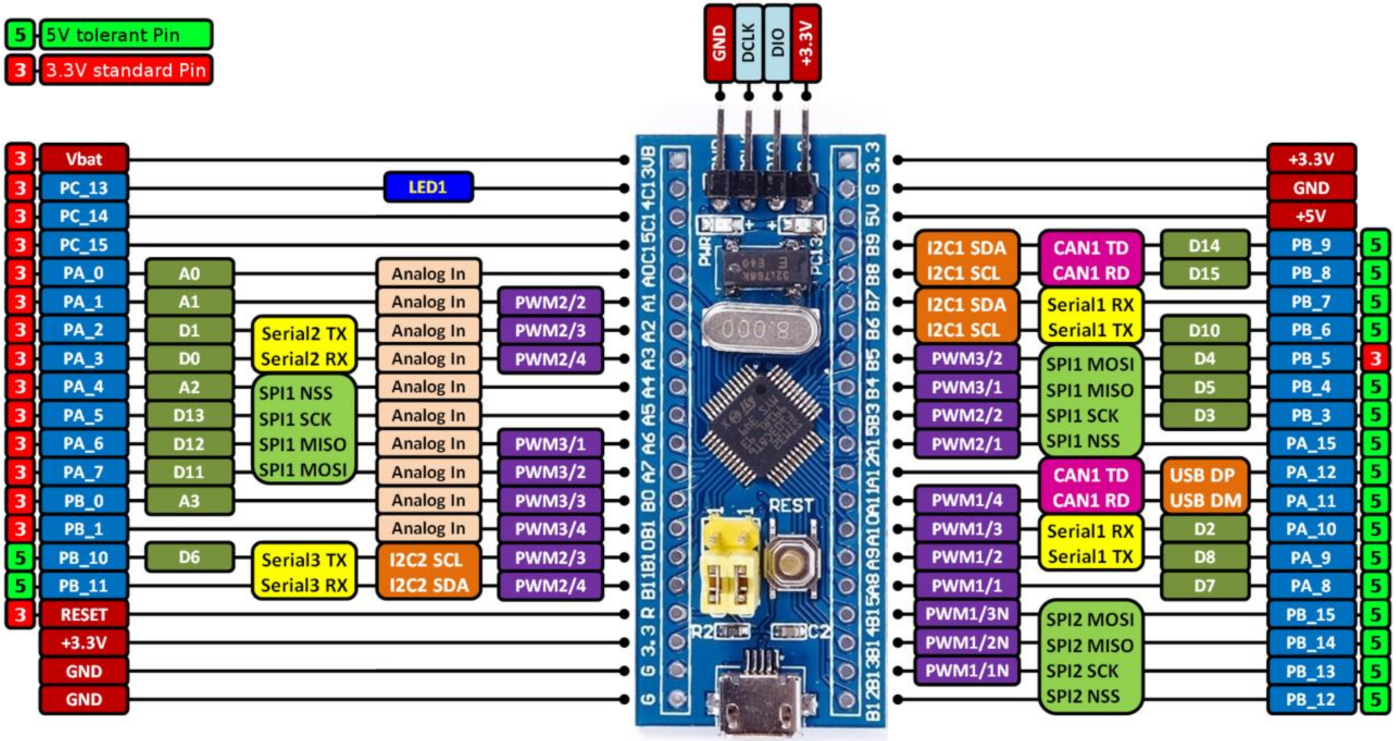
116	C.4	PWML1	D36	
59	C.2	PWML0	D34	
32	D.10		D32	
22	D.9		D30	
16	D.3		D28	
14	D.1		D26	
8	A.15		D24	TF
1	B.26	TCLK0	D22	

7	A.14		D23	TK
13	D.0		D25	
15	D.2		D27	
19	D.6	PWMFI2	D29	
26	A.7		D31	
55	C.1		D33	
60	C.3	PWMH0	D35	
63	C.5	PWMH1	D37	

STOP WARNING! Applying voltages greater than 3.3v to any pin may cause **irreversible damage** to the circuitry on the DUE board, including but not limited to destroying the ARM processor.

GRAY NOMAD
www.robgray.com
8 Jun 2013

Arduino – STM32F103C8T6

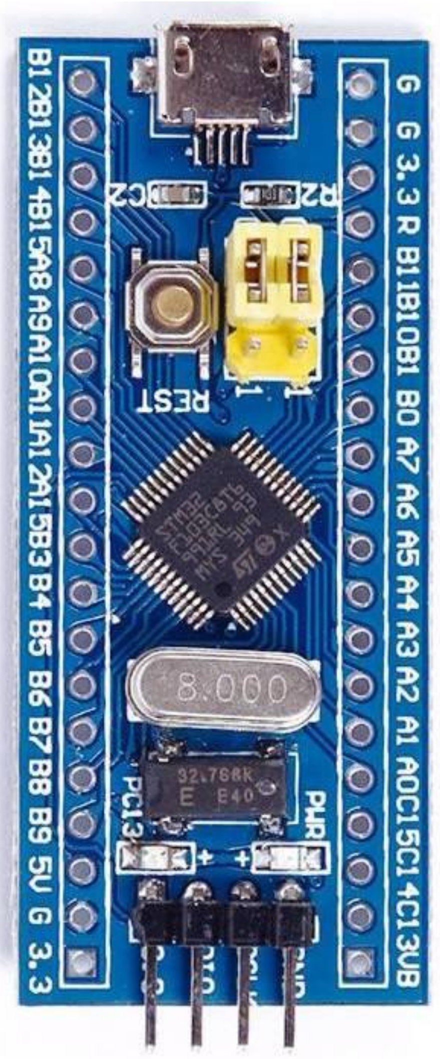


5	5V tolerant Pin
3	3.3V standard Pin

GND DCLK DIO +3.3V

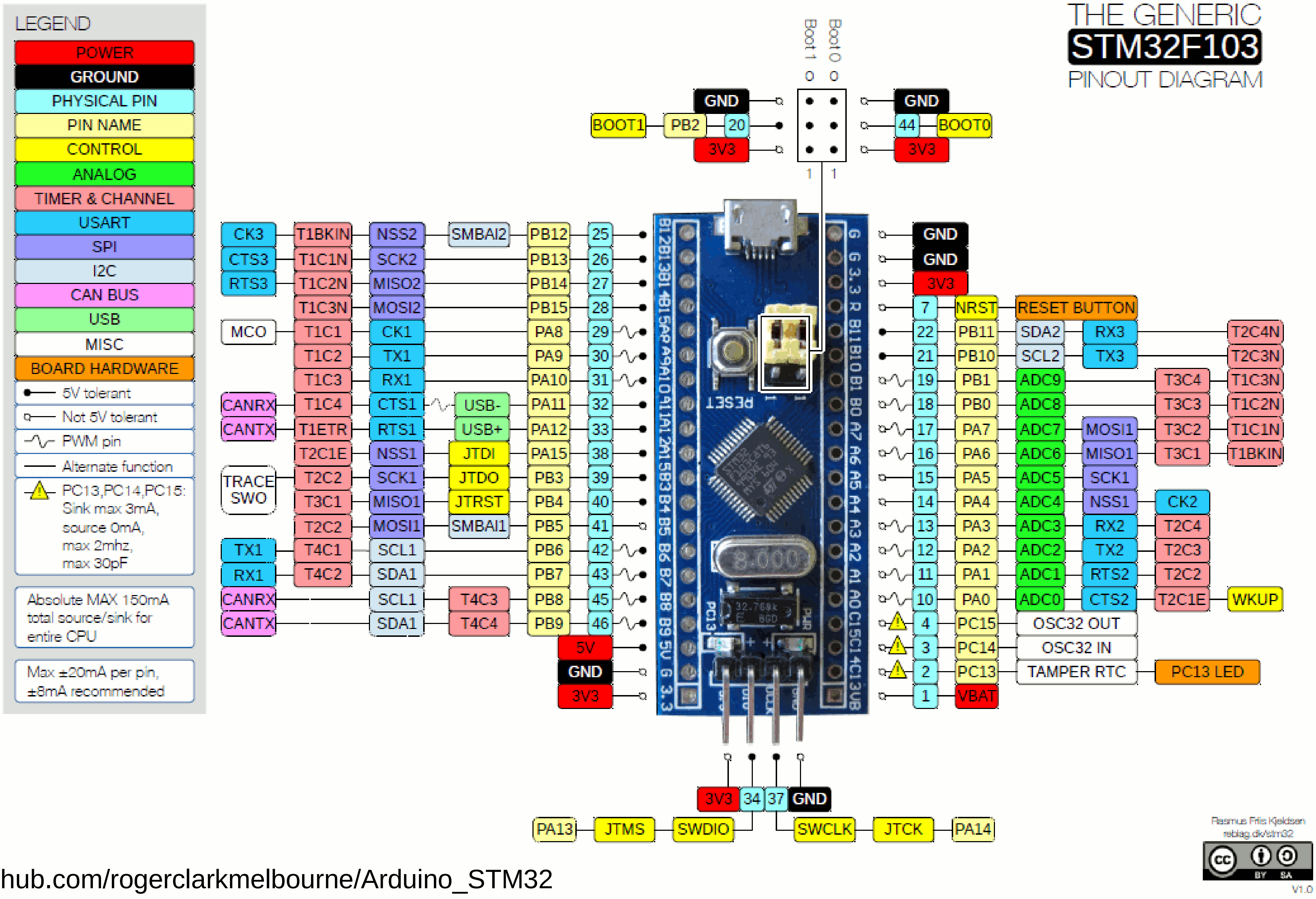
3	Vbat					
3	PC_13			LED1		
3	PC_14					
3	PC_15					
3	PA_0	A0			Analog In	
3	PA_1	A1			Analog In	PWM2/2
3	PA_2	D1	Serial2 TX		Analog In	PWM2/3
3	PA_3	D0	Serial2 RX		Analog In	PWM2/4
3	PA_4	A2	SPI1 NSS		Analog In	
3	PA_5	D13	SPI1 SCK		Analog In	
3	PA_6	D12	SPI1 MISO		Analog In	PWM3/1
3	PA_7	D11	SPI1 MOSI		Analog In	PWM3/2
3	PB_0	A3			Analog In	PWM3/3
3	PB_1				Analog In	PWM3/4
5	PB_10	D6	Serial3 TX	I2C2 SCL		PWM2/3
5	PB_11		Serial3 RX	I2C2 SDA		PWM2/4
3	RESET					
	+3.3V					
	GND					
	GND					

				+3.3V	
				GND	
				+5V	
I2C1 SDA	CAN1 TD	D14		PB_9	5
I2C1 SCL	CAN1 RD	D15		PB_8	5
I2C1 SDA	Serial1 RX			PB_7	5
I2C1 SCL	Serial1 TX	D10		PB_6	5
PWM3/2	SPI1 MOSI	D4		PB_5	3
PWM3/1	SPI1 MISO	D5		PB_4	5
PWM2/2	SPI1 SCK	D3		PB_3	5
PWM2/1	SPI1 NSS			PA_15	5
	CAN1 TD	USB DP		PA_12	5
PWM1/4	CAN1 RD	USB DM		PA_11	5
PWM1/3	Serial1 RX	D2		PA_10	5
PWM1/2	Serial1 TX	D8		PA_9	5
PWM1/1		D7		PA_8	5
PWM1/3N	SPI2 MOSI			PB_15	5
PWM1/2N	SPI2 MISO			PB_14	5
PWM1/1N	SPI2 SCK			PB_13	5
	SPI2 NSS			PB_12	5



github.com/rogerclarkmelbourne/Arduino_STM32

Arduino – STM32F103C8T6



B12 B13 B14 B15 A8 A9 A10 A11 A12 A15 B3 B4 B5 B6 B7 B8 B9 5V G 3.3
G G 3.3 R B11 B10 B1 B0 A7 A6 A5 A4 A3 A2 A1 A0 C15 C14 C13 VB
RESET
8.000
32.768k
PC13
PWR
3.3 DIO DCLK GND

LEGEND

- 🟥 POWER
- ⬛ GROUND
- PHYSICAL PIN
- PIN NAME
- CONTROL
- 🟩 ANALOG
- TIMER & CHANNEL
- USART
- SPI
- I2C
- CAN BUS
- USB
- MISC
- 🟧 BOARD HARDWARE
- ●— 5V tolerant
- ○— Not 5V tolerant
- ∿ PWM pin
- — Alternate function
- ⚠ PC13,PC14,PC15: Sink max 3mA, source 0mA, max 2mhz, max 30pF

Absolute MAX 150mA total source/sink for entire CPU

Max ±20mA per pin, ±8mA recommended

THE GENERIC
STM32F103
PINOUT DIAGRAM

Boot1 ○ Boot0 ○

GND			
BOOT1	PB2	20	●
3V3			

GND		
44	BOOT0	○
3V3		

1 1

CK3	T1BKIN	NSS2	SMBAI2	PB12	25	●
CTS3	T1C1N	SCK2		PB13	26	●
RTS3	T1C2N	MISO2		PB14	27	●
	T1C3N	MOSI2		PB15	28	●
MCO	T1C1	CK1		PA8	29	∿●
	T1C2	TX1		PA9	30	∿●
	T1C3	RX1		PA10	31	∿●
CANRX	T1C4	CTS1	USB-	PA11	32	●
CANTX	T1ETR	RTS1	USB+	PA12	33	●
	T2C1E	NSS1	JTDI	PA15	38	●
TRACE SWO	T2C2	SCK1	JTDO	PB3	39	●
	T3C1	MISO1	JTRST	PB4	40	●
	T2C2	MOSI1	SMBAI1	PB5	41	○
TX1	T4C1	SCL1		PB6	42	∿●
RX1	T4C2	SDA1		PB7	43	∿●
CANRX		SCL1	T4C3	PB8	45	∿●
CANTX		SDA1	T4C4	PB9	46	∿●
				5V		●
				GND		○
				3V3		○

○	GND						
○	GND						
○	3V3						
○	7	NRST	RESET BUTTON				
●	22	PB11	SDA2	RX3			T2C4N
●	21	PB10	SCL2	TX3			T2C3N
○∿	19	PB1	ADC9			T3C4	T1C3N
○∿	18	PB0	ADC8			T3C3	T1C2N
○∿	17	PA7	ADC7	MOSI1		T3C2	T1C1N
○∿	16	PA6	ADC6	MISO1		T3C1	T1BKIN
○	15	PA5	ADC5	SCK1			
○	14	PA4	ADC4	NSS1	CK2		
○∿	13	PA3	ADC3	RX2	T2C4		
○∿	12	PA2	ADC2	TX2	T2C3		
○∿	11	PA1	ADC1	RTS2	T2C2		
○∿	10	PA0	ADC0	CTS2	T2C1E	WKUP	
○⚠	4	PC15	OSC32 OUT				
○⚠	3	PC14	OSC32 IN				
○⚠	2	PC13	TAMPER RTC	PC13 LED			
○	1	VBAT					

○	●	●	○
3V3	34	37	GND

PA13 — JTMS — SWDIO | SWCLK — JTCK — PA14

Rasmus Friis Kjeldsen
reblog.dk/stm32

V1.0

github.com/rogerclarkmelbourne/Arduino_STM32

Arduino – STM32F103C8T6
instalação

Bootloader instalado com FT232RL – Extraído de Roger Clark

- Generic_boot20_PC13.bin

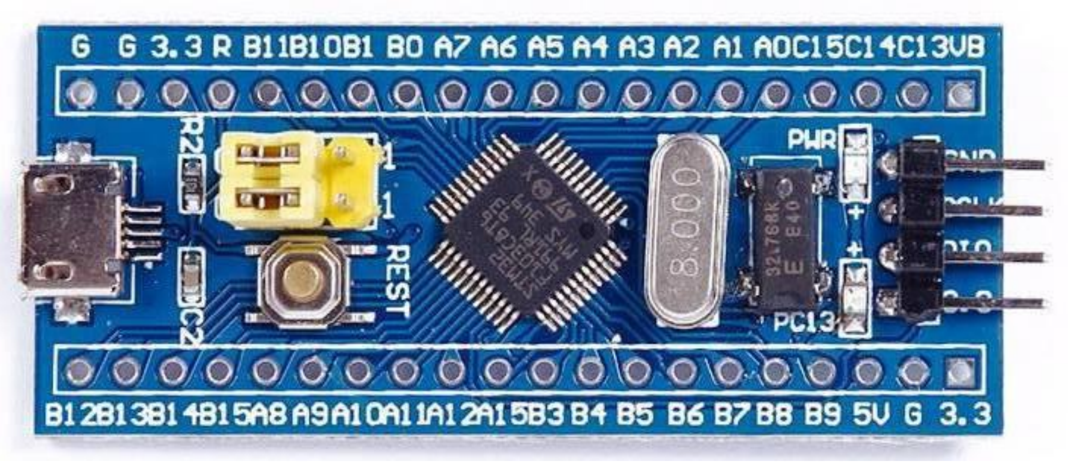
Board na IDE do Arduíno

STM32duino sub-classe

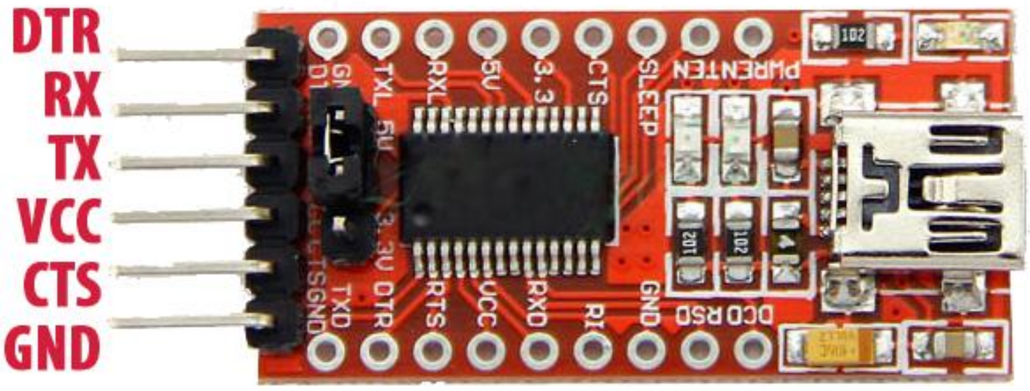
Placa: Generic STM32F103C

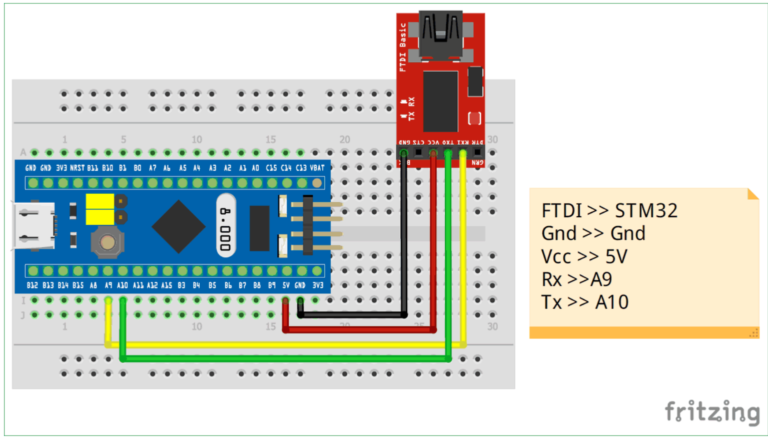
Upload: STM32duino Bootloader

Port: Com6 (Maple Mini)


G G 3.3 R B11B10B1 B0 A7 A6 A5 A4 A3 A2 A1 A0C15C14C13VB
R2
PWR
REST
PC13
C2
B12B13B14B15A8 A9A10A11A12A15B3 B4 B5 B6 B7 B8 B9 5V G 3.3

Arduino – STM32F103C8T6
FT232RL (usb-serial adapter)




FTDI >> STM32
Gnd >> Gnd
Vcc >> 5V
Rx >>A9
Tx >> A10

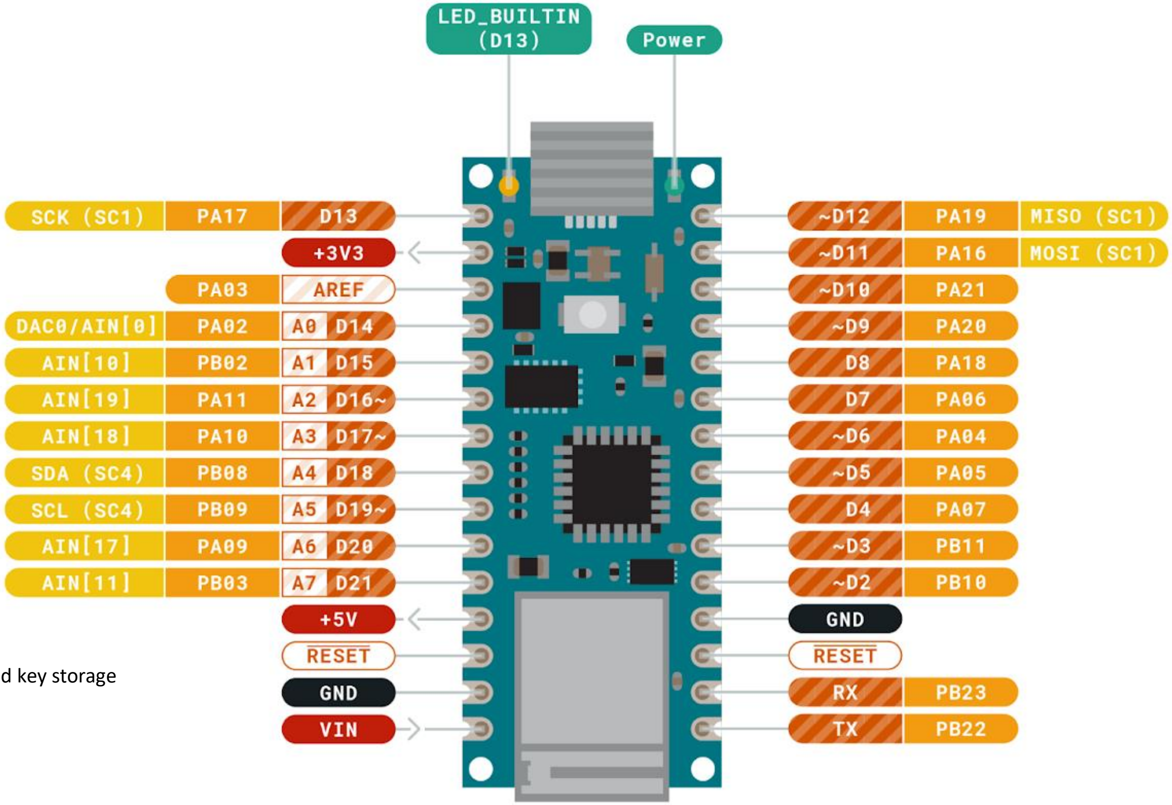
DTR: Data Terminal Ready - an output used for flow control
RX: Serial data Receive pin
TX: Serial data Transmit pin
VCC: Positive voltage output - this is controlled by the jumper. If the jumper is set to 5V, this will provide a 5V output. If the jumper is set to 3.3V, this will provide a 3.3V output.
CTS: Clear To Send - an input used for flow control
GND: Ground or 0V

For most uses, you can simply connect the following pins:
RX on this board to the **TX** pin on your device
TX on this board to the **RX** pin on your device
GND on this board to **GND** on your device

The **VCC** pin is ideal for powering small devices such as homemade circuits. This pin should not be connected when a device has a separate power supply as this may damage both devices.

Please note that in 5V mode the maximum current draw on this pin is approximately 500mA. In 3.3V mode the maximum current draw on **VCC** is approximately 50mA.

Arduino Nano 33 IoT



Nina W102 Module

Dual Core Tensilica LX6 CPU at up to 240MHz

448 KB ROM, 520KB SRAM, 2MB Flash

WiFi

IEEE 802.11b, g, n

2.4 GHz, 13 channels

-96 dBm sensitivity

Bluetooth® BR/EDR

Max 7 peripherals

2.4 GHz, 79 channels

Bluetooth® Low Energy

Bluetooth® 4.2 dual mode

2.4GHz 40 channels

MPM3610 (DC-DC)

Regulates input voltage from up to 21V

More than 85% efficiency @12V

ATECC608A (Crypto Chip)

Cryptographic co-processor with secure hardware based key storage

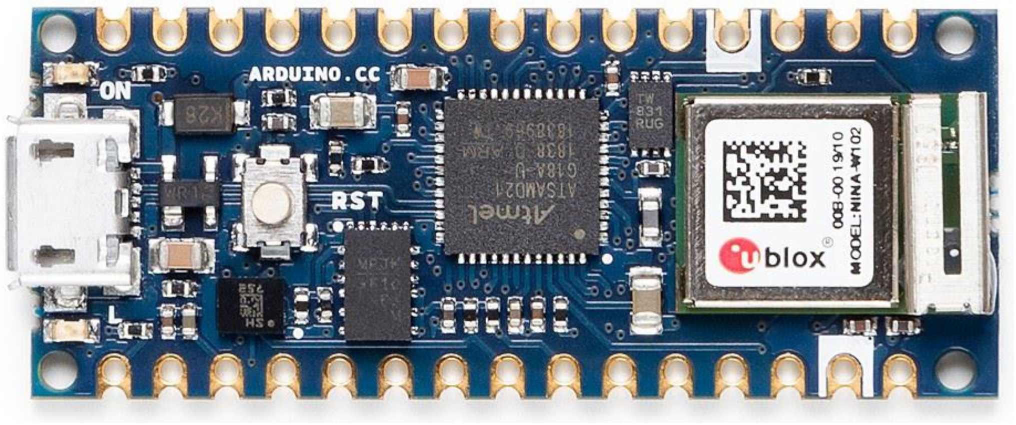
Protected storage for up to 16 keys, certificates or data

LSM6DSL (6 axis IMU)

Always-on 3D accelerometer and 3D gyroscope

Smart FIFO up to 4 KByte based

±2/±4/±8/±16 g full scale



Features

SAMD21G18A Processor

256KB Flash

32KB Flash

Peripherals

12 channel DMA

12 channel event system

5x 16 bit Timer/Counter

3x 24 bit timer/counter with extended functions

32 bit RTC

Watchdog Time

CRC-32 generator

Full speed Host/Device USB with 8 end points

6x SERCOM (USART, I2C, SPI, LIN)

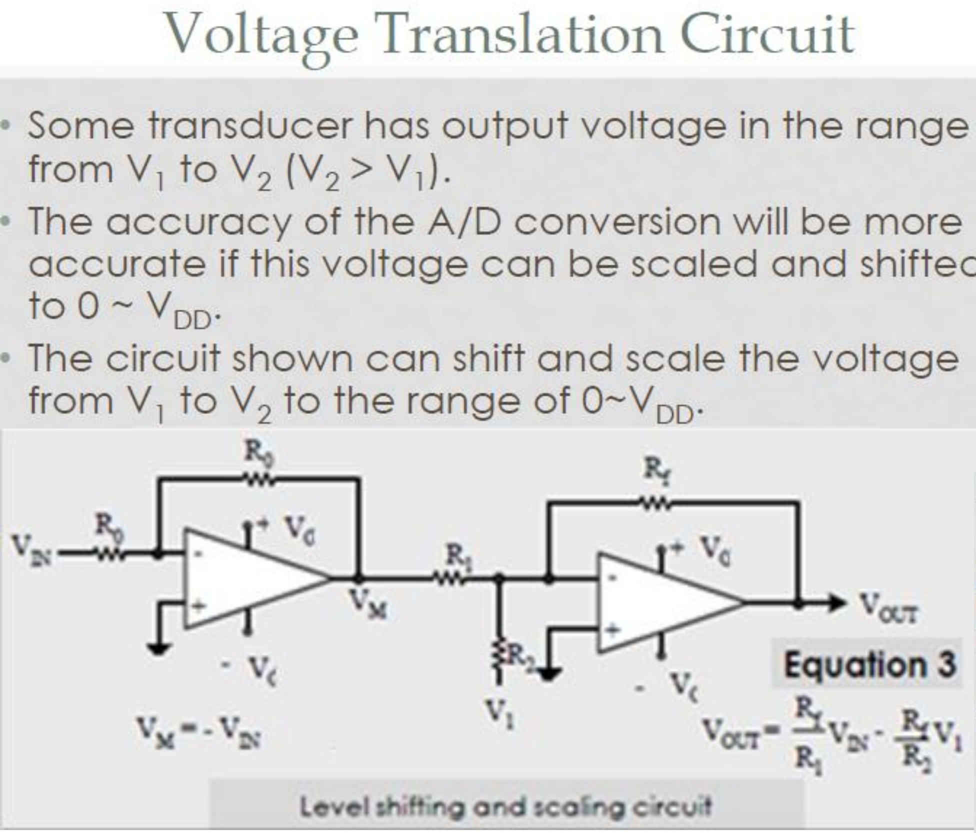
Two channel I2S

12 bit 350ksps ADC (up to 16 bit with oversampling)

10 bit 350ksps DAC

External Interrupt Controller (up to 16 lines)

Arduino – p/ usar o ADC em sinal bipolar



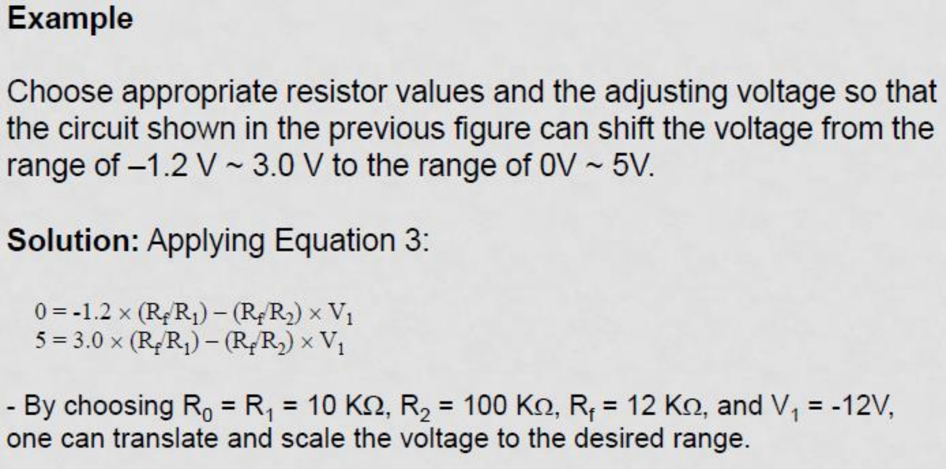
Voltage Translation Circuit

- Some transducer has output voltage in the range from V_1 to V_2 ($V_2 > V_1$).
- The accuracy of the A/D conversion will be more accurate if this voltage can be scaled and shifted to 0 ~ V_{DD}.
- The circuit shown can shift and scale the voltage from V_1 to V_2 to the range of 0~V_{DD}.

Equation 3

$$V_{OUT} = \frac{R_f}{R_1}V_{IN} - \frac{R_f}{R_2}V_1$$

Level shifting and scaling circuit



Example

Choose appropriate resistor values and the adjusting voltage so that the circuit shown in the previous figure can shift the voltage from the range of –1.2 V ~ 3.0 V to the range of 0V ~ 5V.

Solution: Applying Equation 3:

$$0 = -1.2 \times (R_f/R_1) - (R_f/R_2) \times V_1$$
$$5 = 3.0 \times (R_f/R_1) - (R_f/R_2) \times V_1$$

- By choosing $R_0 = R_1 = 10$ KΩ, $R_2 = 100$ KΩ, $R_f = 12$ KΩ, and $V_1 = -12$V, one can translate and scale the voltage to the desired range.

Para V_{in}= -10 a 10 V
V_c= ± 10 V e V_1= -10 V
R_1= R_2 = 4 R_f (usar R_1= ~100 kΩ)